

**SYNCHRONIZATION AND DETECTION FOR TWO-DIMENSIONAL
MAGNETIC RECORDING**

A Dissertation
Presented to
The Academic Faculty

By

Elnaz Banan Sadeghian

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

May 2017

Copyright © Elnaz Banan Sadeghian 2017

SYNCHRONIZATION AND DETECTION FOR TWO-DIMENSIONAL MAGNETIC RECORDING

Approved by:

Dr. John R. Barry, Advisor
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Dr. Steven W. McLaughlin
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Dr. Gordon L. Stüber
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Dr. Matthieu R. Bloch
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Dr. Mary Ann Weitnauer
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Dr. Michael T. Lacey
School of Mathematics
Georgia Institute of Technology

Date Approved: December 9, 2016

To my parents,
Almas and Jalil.

TABLE OF CONTENTS

List of Figures	viii
Summary	xii
Chapter 1: Introduction	1
1.1 The Physics of Magnetic Recording	1
1.2 Two-Dimensional Magnetic Recording	3
1.3 TDMR as a Communication Channel	4
1.4 Components of the Read Channel	7
1.5 Thesis Goal: Synchronization For TDMR	9
1.5.1 Synchronization for Single-Track Detection	9
1.5.2 Synchronization for Multitrack Detection	10
1.6 Summary	10
Chapter 2: Background and Literature Survey	12
2.1 TDMR Models and Detection Strategies	12
2.1.1 Detectors for the 2D ISI Model	13
2.1.2 Detectors for the MIMO Model	14
2.2 Timing Recovery/Synchronization	16
2.2.1 Models of Timing Offset	17

2.3	Synchronization in 1D Magnetic Recording	20
2.3.1	The Basics of Timing Recovery	20
2.3.2	Conventional VCO-Based Timing Recovery	25
2.3.3	Interpolative Timing Recovery	26
2.3.4	Iterative Timing Recovery	27
2.3.5	Per-Survivor Processing (PSP) for Timing Recovery	28
2.4	Synchronization for TDMR	31
2.4.1	2D ISI Model	32
2.4.2	MIMO Model	34
2.5	Summary	36
Chapter 3: Soft Intertrack Interference Cancellation Strategy		38
3.1	Channel Model	38
3.2	Two Detection Strategies	39
3.2.1	Linear Combining	40
3.2.2	Soft ITI Cancellation	41
3.2.3	Numerical Example	46
3.3	Summary	50
Chapter 4: Synchronization For Single-Track Detection		52
4.1	Synchronization Over Separable Channel	52
4.1.1	Asynchronous Linear ITI Suppression	54
4.1.2	Synchronous ITI Cancellation	55
4.1.3	Numerical Example	56

4.2	Synchronization Over Nonseparable Channel	57
4.2.1	Numerical Results	60
4.3	Summary	61
Chapter 5: Synchronization For Multitrack Detection		63
5.1	Channel Model and Assumptions	64
5.2	Detection Algorithms	65
5.2.1	The Case of A Single Isolated Track	65
5.2.2	Joint Detection of Multiple Asynchronous Tracks	71
5.3	Numerical Results	82
5.4	Summary	84
Chapter 6: Equalization to a Time-Varying Target		85
6.1	Channel Model and Assumptions	86
6.2	GPR Equalization Strategies to a Time-Varying Target	87
6.2.1	Prior to Single-Track Detection	87
6.2.2	Prior to Multitrack Detection	94
6.3	Numerical Results	102
6.4	Summary	108
Chapter 7: Conclusion		110
7.1	Contributions	110
7.2	Future Directions	112
7.2.1	Multi-Track Timing Error Detector	112

7.2.2 Media Noise Mitigation for ROTAR	113
Appendix A: Derivation of (3.6) for Linear Case	116
Appendix B: Derivation of (3.16) for Soft ITI Cancellation	117
Appendix C: Joint Optimization of Target and FSE, in single-track detection . .	119
Appendix D: Joint Optimization of Target and FSE, in Multitrack detection . .	122
References	130
Vita	131

LIST OF FIGURES

1.1	A typical HDD for desktop computers (left) [1] can have up to 7 platters. Each platter (right) has a write/read head of it's own which flies very close to the surface of the platter when the platter rotates. Information bits are stored on and read back from concentric circles called data tracks. These tracks are accessed in sections called sectors.	1
1.2	Perpendicular magnetic recording (adapted from [2]). The coil on the write head produces magnetic flux perpendicular to the recording layer. As the write head moves forward on the recording layer, this flux polarizes a few adjacent magnetized regions in one of the two opposite directions to represent binary bits.	2
1.3	Readback signal from perpendicular magnetic recording. The polarity of the polarized region determines the polarity of the signal.	3
1.4	Conventional magnetic recording (left) versus shingled magnetic recording (middle) [6] and the resulted 2D readback waveform from the shingled recording(right).	4
1.5	Magnetic recording as a communication system. In addition to the magnetic medium, in practice, the write and read head also change the ideal modulated signal.	5
1.6	A typical read channel for magnetic recording consists of timing recovery, equalization, and detection blocks.	7
2.1	Two approaches to TDMR detection: 1) 2D ISI model (left), versus 2) MIMO model (right).	14
2.2	Real vs. expected readback waveform. The solid waveform is the real readback waveform where the pulses arrive at times $\{kT + \tau_k\}$. However, the read channel expects to receive the dashed waveform where the pulses arrive at ADC sampling times $\{kT\}$	17

2.3	Models of timing offset in magnetic recording channel.	19
2.4	Models of timing offset in magnetic recording channel.	20
2.5	Basic timing recovery.	22
2.6	A conventional VCO-based timing recovery.	25
2.7	Interpolated timing recovery in 1D magnetic recording. This figure is adapted from [4] with an added equalizer after the interpolation filter (resampler).	26
2.8	(a) Read channel with classical timing recovery, vs. (b) iterative timing recovery.	28
2.9	The read channel with per-survivor processing for timing recovery in 1D magnetic recording.	29
2.10	Viterbi algorithm with PSP-based timing recovery on PR-IV trellis.	33
2.11	A MIMO model with $K = 2$ asynchronous tracks and $N = 2$ overlapping read heads. (The asynchronous between bit rates are exaggerated for effect.)	34
3.1	Iterative detector based on turbo equalization [36].	47
3.2	FER performance in the absence of media noise ($\gamma = 0$).	49
3.3	FER performance when media noise is dominant ($\gamma = 0.8$).	50
4.1	Asynchronous linear ITI suppression	54
4.2	Synchronous ITI cancellation	55
4.3	BER performance of synchronous ITI cancellation detector (the same as that of asynchronous ITI suppression) in the presence and absence of timing offsets (for convenience we renumber the tracks: track 0 is the middle track and tracks ± 1 are its neighbors).	56
4.4	Readers positions relative to the data tracks are marked with dashed lines. The two readers (indicated with their responses) were used to extract the channel impulse response. The tracks width (pitch) is $22.1nm$, and the reader width is 85% of the track pitch.	58

4.5	The proposed architecture for detecting the middle track: (a) the MIMO model with frequency offset of a nonseparable channel with $N=2$ readers and $K = 3$ tracks, (b) a MISO equalizer to detect the middle track, and (c) a PSP algorithm embedded inside a Viterbi detector	59
4.6	BER performance of the proposed architecture of Fig. 4.5 for detecting the center track.	61
5.1	An example of two tracks of interest whose timing differ in frequency and phase, and two readers with significant overlap.	64
5.2	Conventional modular (lower branch) versus alternative (upper branch) strategy in synchronization and detection of a single isolated track.	67
5.3	An illustration of a moving target for the case of frequency offset with $\Delta T/T = 2 \times 10^{-4}$, and a sector length $L = 10^4$, assuming $M = 8$: (a) The target $\mathbf{h}[0]$ at time $k = 0$; (b) the target $\mathbf{h}[2500]$ at one quarter of the sector; (c) the target $\mathbf{h}[7500]$ at three quarters of the sector; and (d) the target $\mathbf{h}[10000]$ at the end of the sector.	68
5.4	BER performance of the alternative strategy with time-varying target based on $h = [1, 0.5]$, in detecting a single isolated track from a single readback waveform of (5.9).	70
5.5	BER performance of the alternative strategy with time-varying target based on (5.12) in detecting $K = 2$ tracks from $N = 2$ readback waveform.	73
5.6	BER performance of the ROTAR algorithm with PSP.	83
6.1	An example of two tracks of interest whose timing differ in frequency and phase, and two readers with significant overlap.	87
6.2	GPR equalization strategies, for the exampling given in Fig. 6.1 and detecting track 2: (a) conventional strategy, (b) alternative strategy 1, and (c) alternative strategy 2.	88
6.3	A parametric illustration of the alternative equalization strategy 1 for the case of detecting a single track from $N = 2$ waveforms as shown in Fig. 6.2b.	90
6.4	A parametric illustration of the alternative equalization strategy 2 for the case of detecting a single track from $N = 2$ waveforms as shown in Fig. 6.2c.	93

6.5	A parametric illustration of the alternative strategy 2 for the case of jointly detecting $K = 2$ tracks from $N = 2$ waveforms as shown in Fig. 6.1.	95
6.6	The description of DSI data set. 25 different reader positions separated by $TP/8$ are marked with dashed lines. The resulted 25 readback waveforms mainly cover the three middle tracks. The data is generated from the grain-flipping probability model. A write timing frequency offset of $\Delta T_2 = 2 \times 10^{-4}$ is injected into tracks 2 while all other tracks are written synchronously to each other. As shown, we used two reader positions in the simulations to detect tracks 1 and 2.	102
6.7	The alternative strategy 2 for equalizing $N = 2$ readback waveforms shown in Fig. 6.6 covering track 1 with $\Delta T_1 = 0$ and track 2 with $\Delta T_2/T = 2 \times 10^{-4}$. 103	
6.8	The PLL performance in estimating the timing offset of track 2.	104
6.9	Convergence of the LMS versus RLS algorithms when used in the alternative strategy 2 of Fig. 6.7. MSE is averaged over the past 100 samples.	105
6.10	The MIMO equalizer output $y_k^{(2)}$ versus the corresponding signal $d_k^{(2)}$ from the target branch in Fig. 6.7.	106
6.11	The proposed read channel for jointly detecting the two tracks of Fig. 6.1. .	107
6.12	The BER performance of the proposed read channel.	107
C.1	Joint optimization of target and equalizer for the case of $N = 3$ readers and $K = 1$ track of interest.	119
D.1	Joint optimization of target and equalizer for the case of $N = 2$ readers and $K = 2$ tracks of interest.	123

SUMMARY

Two-dimensional magnetic recording (TDMR) is a new recording architecture that supports a drastic increase in the data density on conventional magnetic recording hard disk drives. The gain from TDMR comes from two directions: The shingled writing, where the adjacent data tracks are written with partial overlap on top of each other in order to squeeze many more number of tracks on the disk and increase the data density, and also from powerful signal processing algorithms that enable efficient data recovery from highly interfered and noisy read back signals.

This thesis develops synchronization and detection algorithms for current and future generations of TDMR channel. In the current generation of TDMR, multiple readers are used to detect a single data track at a time. A naive read channel would first perform timing recovery separately on every readback waveform received, and then it would equalize and detect a track of interest. Therefore, the number of synchronization blocks needed would be the same as the number of readback waveforms. This thesis proposes a new read channel where equalization *precedes* timing recovery and detection. Consequently, the number of synchronization blocks reduces to only *one* for every track being detected, and thereby the proposed read channel significantly reduces the computational complexity of a conventional read channel.

To achieve the full potential of TDMR, future generations of TDMR read channels, however, will detect multiple tracks at a time. Multi-track detection utterly changes the synchronization problem since adjacent tracks can have slightly different bit rates. The new challenge, therefore, is the impossibility of simultaneously synchronizing the ADC sampling times to multiple rates. In this context, synchronization can no longer be performed as a separate block in the conventional fashion. This thesis proposes rotating-target (ROTAR) algorithm as a first solution for joint detection of multiple asynchronous tracks from one or more readback waveforms. ROTAR jointly performs the synchronization and detection tasks

using a Viterbi detector based on a time-varying target that results when the asynchrony of the tracks is absorbed into the underlying target. ROTAR also uses per-survivor processing for estimating the unknown timings.

Further, this thesis completes a proposed read channel for future generations of TDMR by proposing an equalization strategy to precede the ROTAR detector.

CHAPTER 1

INTRODUCTION

Data-driven companies such as Google, Facebook, Yahoo, and Netflix rely on data storage facilities to store and retrieve their data. Today, two of the most important data storage/retrieval devices are hard disk drives (HDD) and solid state drives (SSD). HDD's are generally cheaper and can store larger volumes of data. In contrast, SSD's are much smaller, and faster in writing the data and reading the data back. For this reason, SSD's dominate the personal computer market, while HDD's dominate the data center and large-capacity data infrastructure markets.

1.1 The Physics of Magnetic Recording

Fig. 1.1 shows a typical HDD for desktop computers. Main components of a typical HDD consists of up to 7 circular disks or platters that store the data, one write/read head mounted on the tip of an actuator arm, one for each platter, a DSP chip with the firmware that controls the operation of all the parts, and most importantly, signal processing software for recovering

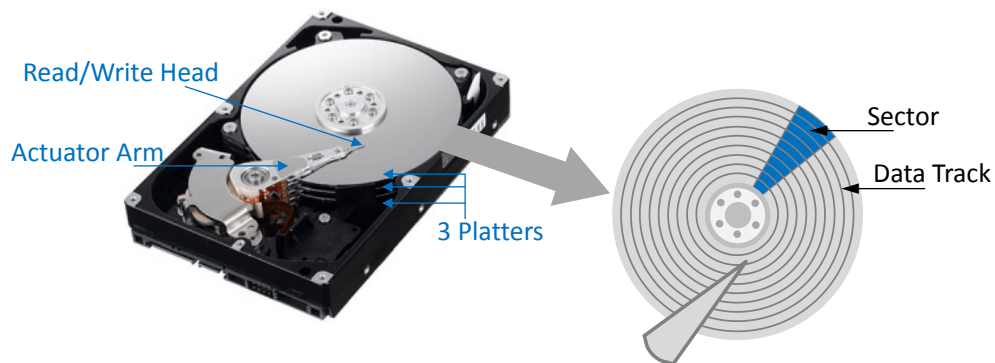


Figure 1.1: A typical HDD for desktop computers (left) [1] can have up to 7 platters. Each platter (right) has a write/read head of it's own which flies very close to the surface of the platter when the platter rotates. Information bits are stored on and read back from concentric circles called data tracks. These tracks are accessed in sections called sectors.

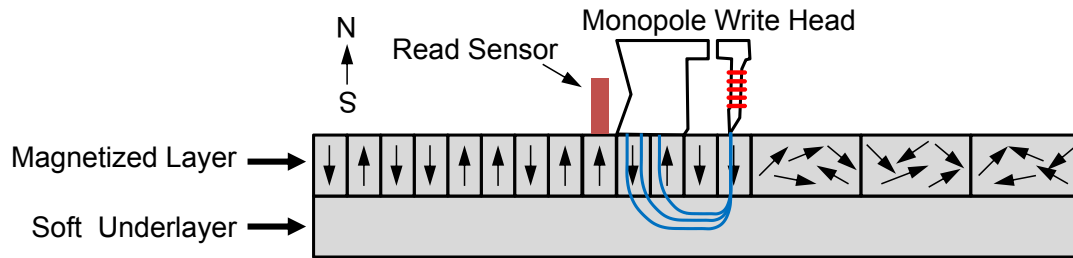


Figure 1.2: Perpendicular magnetic recording (adapted from [2]). The coil on the write head produces magnetic flux perpendicular to the recording layer. As the write head moves forward on the recording layer, this flux polarizes a few adjacent magnetized regions in one of the two opposite directions to represent binary bits.

the information bits from the signals received from the read head.

Each platter is coated by a thin film of ferromagnetic material that can be polarized to represent binary information bits. Depending on the direction of this polarization relative to the surface of the ferromagnetic layer, we can have longitudinal or perpendicular magnetic recording. Since each polarized region in perpendicular magnetic recording takes much less area of the surface of the ferromagnetic layer compared to the longitudinal recording, today, longitudinal magnetic recording is obsolete and only perpendicular recording is implemented.

Fig. 1.2 shows perpendicular magnetic recording (adopted from [2]). The coil on the write head produces a magnetic flux that is perpendicular to the ferromagnetic layer. The write head records the information bits by polarizing a few adjacent magnetized regions (also known as magnetic grains) on the recording layer in one of the two opposite directions. As the write head moves along the recording layer, each bit is represented by a group of adjacent grains with the same polarity. Fig. 1.3 displays a readback signal from the read head sensor. The sensor detects the magnetization of the bit regions as the disk rotates causing the readhead to scan the recorded data.

The main objective of the magnetic recording industry is to increase the areal density, as measured by the number of stored bits per unit surface area of the disks. According to the roadmap released by advanced storage technology consortium [3], the highest areal density in today's drives is about 1.3 Terabits per square inch of the medium, and is going to achieve

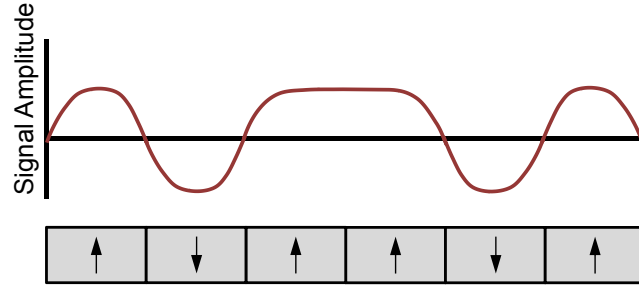


Figure 1.3: Readback signal from perpendicular magnetic recording. The polarity of the polarized region determines the polarity of the signal.

10 Terabits per square inch in 2025. One obvious way to increase the areal density is to shrink the size of the magnetic grains that represent each bit. However, the magnetic grains can only be shrunk to the point where they can be thermally unstable and lose their polarity. This phenomenon is called the *superparamagnetic limit* which prevents the increase in the areal density after some point [4]. To overcome this limit, there has been many attempts both in changing and improving the magnetic medium itself and also in the recording technology as a whole: heat-assisted magnetic recording (HAMR), bit-patterned magnetic recording (BPMR), and more recently two-dimensional magnetic recording (TDMR) [5].

1.2 Two-Dimensional Magnetic Recording

Since 2010, the disk drive industry is pursuing a huge increase in the areal density up to 10 Terabits per square inch of the medium through two-dimensional magnetic recording technology [6]. TDMR refers to the combination of shingled magnetic recording and data detection based on multiple readback waveforms.

Fig. 1.4 (left) illustrates the conventional magnetic recording when the data tracks are written side by side with some guard space in between to avoid interference between adjacent tracks. To achieve higher densities, shingled magnetic recording (Fig. 1.4 (middle)) shrinks the guard space between the tracks allowing the tracks to overlap one another, like roofing shingles. Although data is written with the same large write head, a narrower track is all that remains after shingling. By allowing tracks to overlap, areal density can continue

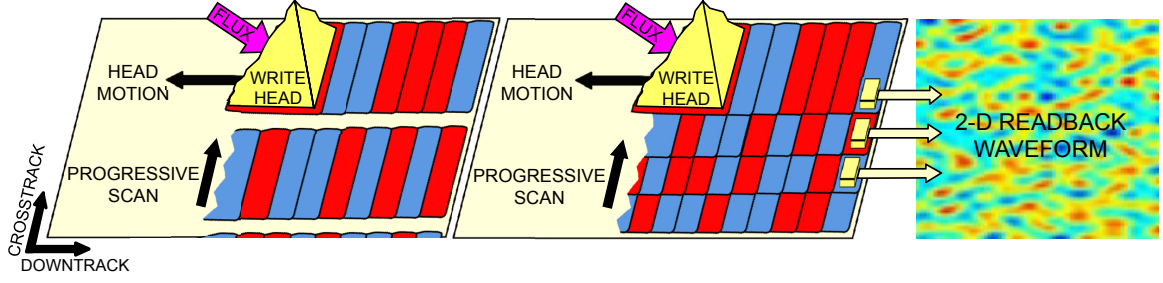


Figure 1.4: Conventional magnetic recording (left) versus shingled magnetic recording (middle) [6] and the resulted 2D readback waveform from the shingled recording(right).

to scale without further shrinking the size of the write heads.

1.3 TDMR as a Communication Channel

The magnetic recording, regardless of HAMR, BPMR, and TDMR technologies, is modeled as a communication system: The transmitter encodes, modulates, and writes the information bits on the magnetic medium, the receiver reads and recovers the information bits from the magnetic medium. Fig. 1.5 models the magnetic recording as a communication system. Since in addition to the magnetic medium, the write and read head also changes the perfect modulated signal, it is too a part of the communication channel. In this communication system, the time replaces the position: The information is written and later read back at another time instead of at another place. Similar to any communication system, the information bits are error-correction coded and modulated into a continuous-time signal. This signal is a current that runs through the coil in the write head producing the magnetizing flux. The readhead senses the polarized regions and outputs a voltage signal, as illustrated in Fig. 1.3. The entire signal processing unit that estimates the information bits from the readback signal is referred to as the *read channel*.

In any communication system, ideally, we expect to receive the same signal we had transmitted. In magnetic recording, we also expect to readback a signal very similar to the current signal that wrote the bits on the medium. However, as it is the case for all

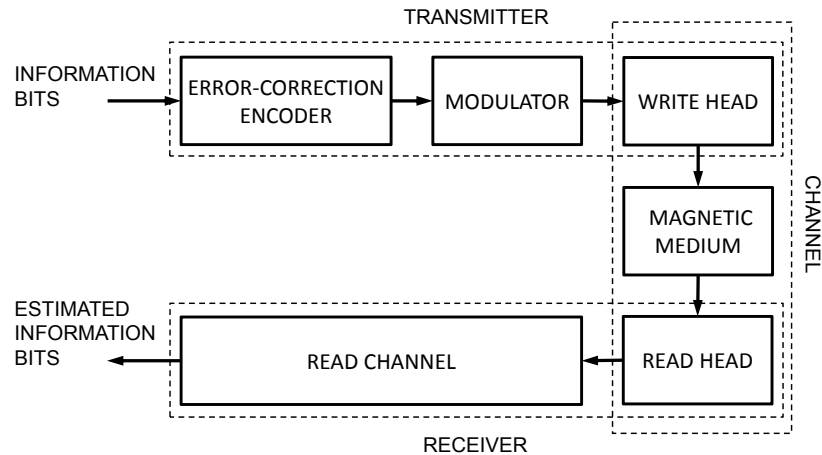


Figure 1.5: Magnetic recording as a communication system. In addition to the magnetic medium, in practice, the write and read head also change the ideal modulated signal.

communication systems, there is the channel effect: an unfortunate series of changes to the original signal that make the data recovery a challenge. The channel effect in magnetic recording include:

1. Intersymbol interference (ISI), in general, is the blurring of adjacent bits together which makes the recovery of each individual bit difficult. In magnetic recording, ISI refers to the blurring of the adjacent bits on the same data track, and not between the adjacent tracks. ISI exists because the channel is bandlimited. A bandlimited channel changes the shape of the pulse shape that arrives at the receiver. The write signal is a perfectly rectangular signal. The readback signal, however, is more or less similar to a superposition of shifted Gaussian pulses. Since the channel has a cut-off frequency, each received pulse shape in the time-domain will be a never-ending pulse which interferes with adjacent pulses and causes ISI.
2. Intertrack interference (ITI) is the blurring of the bits on adjacent tracks in magnetic recording, similar to crosstalk in wireless communication. ITI is more severe in TDMR technology. In fact, as illustrated in Fig. 1.4, the shrinkage of the track widths gives rise to ITI in the crosstrack dimension. The shrinkage can continue to increase the areal density to such an extent that ITI becomes as severe as the ISI in

the downtrack dimension. For this reason, continuous increase in the areal density through shingled writing results in combination of ISI plus ITI in the two dimensions and that explains the term *2D magnetic recording*.

3. Media noise, as the name suggests, is the noise generated by the magnetic medium. Fig. 1.3 seems to suggest that the impulse response, from one magnetic grain to the next, does not change and the readback signal is only a superposition of shifted identical impulse responses. To the contrary, the shape of the impulse response changes from one magnetic grain to the next. The reason is that the magnetic grains have irregular shapes. Therefore, bit regions will also have irregular and random shapes. This irregularity changes the shape of the impulse response from one bit to the next. The effect is that the impulse response will *jitter* from one bit to the next, leading to an inaccuracy in the superposition signal of Fig. 1.3. This inaccuracy is known as the *media* or *jitter* noise, which makes up a predominate portion of the total noise. Media noise, however, is colored and data dependent, which can be exploited by the read channel to reduce its impact.
4. Electronic noise is the well-known additive white Gaussian noise (AWGN) that is common to almost all communication channels.
5. Position uncertainty: Fig. 1.4 (right) illustrates a typical readback waveform from TDMR. Here, the 2D interference along with several other impediments of the magnetic medium demand advanced signal processing strategies of manageable complexity that are able to extract the user bits from a readback waveform similar to Fig. 1.4 (right). the problem is that the positions of the bits on a readback waveform are not exactly known. Knowing *where* the bits are, however, is a prerequisite to knowing *what* the bits are. The mismatch between the write and read clock frequencies and phases, or, in another words, the mismatch between the actual positions of the bits and the times in which the analog-to-digital converter (ADC) samples the signal

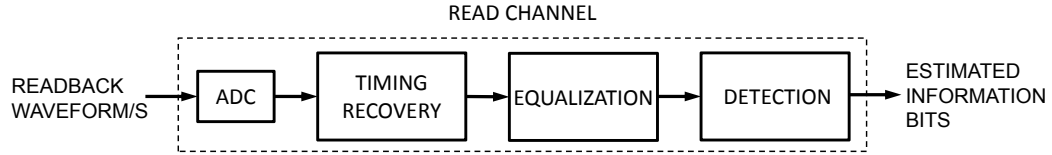


Figure 1.6: A typical read channel for magnetic recording consists of timing recovery, equalization, and detection blocks.

creates this position uncertainty. *Timing recovery* or *synchronization* solves with position uncertainty. Synchronization is an essential component of the state-of-the-art read channel design for TDMR. It refers, in general, to a part of the read channel that compensates for the asynchrony between the ADC sampling times and the desired sampling times, as dictated by the positions of the bits.

1.4 Components of the Read Channel

As mentioned above, the TDMR channel affects the write signal by introducing position mismatch, ISI, ITI, media, and electronic noise. A readback signal, bearing all those adverse effects, is what is received as the input to the read channel. The read channel is the collective signal processing blocks that aim at mitigating those effects for an optimal detection and recovering the written bits. A typical read channel block diagram is shown in Fig. 1.6. Here, we provide a brief introduction of its signal processing components:

1. **Timing Recovery:** The received readback signal is inherently continuous in time. For an efficient and therefore digital read channel implementation, the very first step is an ADC block. However, in practice, the ADC sampling times are not exactly aligned with the positions of the bits. Therefore, the ADC output samples are not the ideal samples for detecting the information bits. The timing recovery block receives these misaligned samples and outputs a best possible version of the ideal aligned samples that would have resulted if the ADC sampling times were exactly aligned with the positions of the bits.

2. Equalization: In magnetic recording, the blurring effect between adjacent bits spreads over tens to hundreds of the adjacent bits in the downtrack dimension and over a few bits in the crosstrack dimension. Continuous shrinkage of the track widths in TDMR, however, can potentially spread the ITI in crosstrack dimension to the same margins as the ISI in the downtrack dimension. Equalization refers to the signal processing strategies that try to mitigate this blurring effect. In general, an equalizer tries to either shorten the extent, or to reduce the severity, or to completely remove this fusion of the adjacent bits. A zero-forcing (full-response) equalizer attempts to completely remove the blurring by filtering the received signal with the inverse of the channel frequency response, such that the overall result is a flat response throughout the frequency domain. This is too extreme to be applicable. In part because even if the channel response has finite length, the inverse response might be infinitely long. And in part because the inverse response will have a large magnitude at those frequencies where the channel response is weak. As a consequence, the equalizer boosts any noise that comes after the channel at those frequencies and thereby destroys the overall signal-to-noise ratio. A more balanced equalization does not equalize the signal all the way to the delta function response like the *full-response* equalizer does. It only partially removes the interference by shortening its extent and/or its severity. The *partial response* (PR) equalization is a technique that equalizes the channel output to the output of another fictitious channel that is significantly shorter in the length and that causes less interference. This fictitious channel response is called the *target*.
3. Detection: A detector is an algorithm that receives a noisy train of modulating pulse shapes which are partially blurred with other neighboring pulse shapes, and outputs an estimate of the original information bits. Since both the noise and the written bits are widely modeled as stochastic variables, the received discrete-time signal can be viewed as the output of an stochastic process, where probabilistic detection is optimal. Probabilistic detectors include maximum a-posteriori probability (MAP)

and maximum likelihood (ML) detectors. MAP detectors are the optimal detectors since they minimize the probability of detection error. They reduce to ML detectors whenever the written bits are equiprobable. For ISI channels with additive white Gaussian noise (AWGN), a practical ML sequence detector is implemented through Viterbi algorithm [7]. This algorithm finds *the most-likely sequence of the written bits*. Also, the MAP detector is implemented with BCJR algorithm [8] that finds *the sequence of the most-likely written bits*. Further, since the bits are error-correction coded, the detector also includes a decoder for detection of the original uncoded bits.

1.5 Thesis Goal: Synchronization For TDMR

The TDMR literature includes countless state-of-the-art read channel designs that focus on one or a combination of several blocks of the read channel, among which synchronization has received less attention. This section discusses the goal of this thesis that is to develop synchronization strategies for TDMR.

1.5.1 Synchronization for Single-Track Detection

Current implementations of read channels for TDMR detect one track at a time. Single-track detection refers to detecting one track of interest at a time, using one or more readback signals that can result from a single pass of an array of read heads or multiple passes of a single read head. Single-track detection has been the norm from the very beginning, starting with the first hard disk drives in the late 1950's, up until today's TDMR implementations with multiple readers. In this setting ITI is a nuisance that should be avoided. The read channel designs for single-track detection in TDMR try to suppress ITI prior to the detection. As a result, the detection problem for TDMR is reduced to the conventional 1D detection problem of 1D magnetic recording. The benefit is that since the existing detection and synchronization strategies for 1D magnetic recording are matured, here, they can be leveraged with no additional cost.

In this thesis we improve on 1D synchronization and detection strategies. In particular we suggest alternative strategies in which the implementation cost of a typical read channel is greatly reduced.

1.5.2 Synchronization for Multitrack Detection

Multitrack detection refers to a joint detection of multiple adjacent tracks, using one or more readback waveforms. We expect a huge increase in the areal density as a result of multitrack detection *that embrace ITI* rather than *avoid ITI*, similar to the increase in the areal density that was achieved when partial response maximum-likelihood (PRML) strategies *that embrace ISI* replaced peak detection strategies that *avoid ISI*. For this reason, future implementations of TDMR will jointly detect multiple tracks [5], where the synchronization problem will fundamentally change from its conventional 1D setting. As it will be detailed in Chapter 5, the synchronization can no longer be performed separately. Rather, it must be performed jointly within the detection.

In multitrack detection setting, unlike the conventional single-track detection, there has been no prior published work that addresses the synchronization problem. In this thesis, we propose a rotating-target (ROTAR) algorithm for jointly detecting multiple asynchronous tracks from one or more readback waveforms.

1.6 Summary

We presented a brief introduction to the magnetic recording and TDMR technology. Magnetic recording, in general, and TDMR, in particular, are modeled as a communication system where the transmitter *writes* the data on the magnetic medium, the channel, and the receiver *reads back* and recovers the data from the channel at a later time. We itemized the channeling effects, the detection impediments of TDMR channel. We, also established that synchronization is a *prerequisite* to the detection problem. Further, we classified the entire detection and synchronization problem for TDMR into two categories: 1) the

synchronization and detection for the past and the current implementations of TDMR, where data tracks are detected on a one-by-one basis, and 2) the synchronization and detection for the future implementations of TDMR, where data tracks are detected jointly. We established the objective of the thesis to 1) improve upon the former category through reduction in computational complexity, and 2) to devise a solution for synchronization and detection for the latter category.

CHAPTER 2

BACKGROUND AND LITERATURE SURVEY

In the magnetic recording literature, a detection strategy refers to the entire read channel architecture aimed at detecting the information bits from the received waveform or waveforms. In this chapter we first present a literature survey on TDMR detection strategies which fall into a clear bifurcation based on the approach taken in modeling the TDMR channel. Next, we focus on the problem of timing recovery and provide the necessary background information and fundamental concepts that enable us to follow the synchronization and detection algorithms of Chapters 4, 5, and 6. We further discuss the timing recovery strategies that have been employed or proposed for 1D magnetic recording and also for TDMR so far.

2.1 TDMR Models and Detection Strategies

There exist two distinct discrete-time channel models in TDMR prior art:

1. The *two-dimensional (2D) ISI* model, in which the written bits and the readback waveforms are modeled as 2D signals with two interchangeable dimensions.
2. The *multiple-input multiple-output (MIMO)* model, in which the written bits and the readback waveforms are modeled as vector-valued functions of time.

The choice of which model to use depends on the number N of available readback waveforms. We denote $N = N_p N_r$ where an array of N_r readers make N_p passes over different regions of a disk. The MIMO model is appropriate if N is relatively small compared to the length of the waveforms. The 2D ISI is appropriate if N is large and comparable in size to the length of the waveforms. The MIMO model suits low-latency applications which

cannot tolerate the delay caused by multiple passes of the read head array, whereas the 2D model is appropriate for applications which can afford a scan over the entire surface of the disk.

2.1.1 Detectors for the 2D ISI Model

The **2D model** arises when the total number N of available readback waveforms is comparable in size to the length of the waveforms. Therefore, the readback waveforms are represented as a 2D signal (matrix) with the two dimensions comparable in size and interchangeable. The channel impulse response is also a matrix which captures both ITI in the crosstrack dimension and ISI in the downtrack dimension. The 2D model is obtained using a 2D convolution of the channel impulse response with a large matrix of recorded bits. An example of a 2D signal obtained from a 2D convolution is shown in Fig. 2.1 (left).

Detector design for the 2D channel is an active area of research. The maximum-likelihood (ML) detector for the 2D ISI channel is prohibitively complex; there is no 2D analog of the Viterbi detector that enables optimal performance with low complexity [9]. A variety of sub-optimal detectors with reduced complexity have been proposed, many of which can be viewed as 2D extensions of 1D detectors. The detector in [10], for example, uses four 1D decision-feedback equalizers (DFEs) to scan the 2D signal in four different directions. The detector in [11] achieves near-ML performance by iterating between Bahl, Cocke, Jelinek and Raviv (BCJR) [8] equalizer for the rows and DFE for the columns of the 2D signal. The detector of [12] iterates between a binary and a non-binary BCJR detector, respectively, for the rows and the columns of a coded 2D signal on a separable 2D ISI channel; this detector falls only 1 dB short of an interference-free channel and thereby encourages equalizing a general 2D impulse response to a nearby (in MMSE sense) separable matrix. A generalized belief propagation detector is proposed in [13] that exploits the data-dependent media noise. Several other detectors have been proposed based on iterative decision feedback, in which extrinsic information is exchanged in a turbo fashion

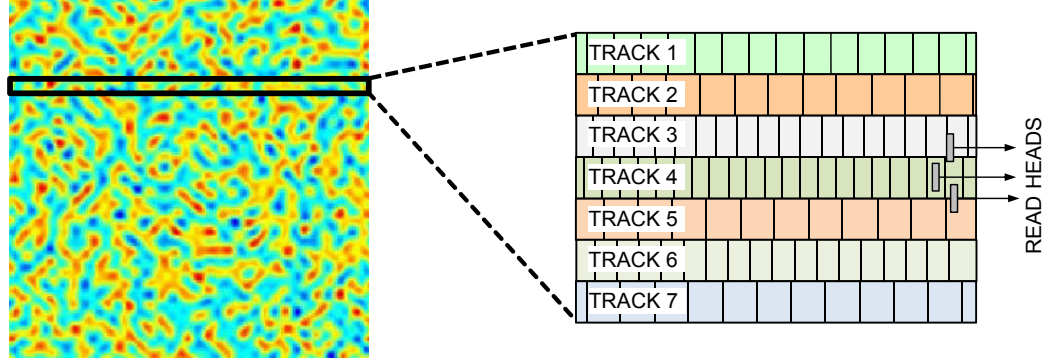


Figure 2.1: Two approaches to TDMR detection: 1) 2D ISI model (left), versus 2) MIMO model (right).

between modified BCJR detectors, including the row and column soft decision feedback algorithm [14], iterative soft decision zig-zag algorithm [15], and a multi-row/column detector coupled with a 2D equalizer [16].

2.1.2 Detectors for the MIMO Model

The **MIMO model** arises when the number N of available readback waveforms is small. In practice, N can be as small as 2 readback waveforms collected from a single pass of an array of $N_r = 2$ readers. Nevertheless, as Fig. 2.1 depicts, a connection between the two models can be made: Assume a 2D readback waveform of Fig. 2.1 (left). The MIMO model of Fig. 2.1 (right) can be obtained by discarding all but $N' = 3$ rows of the 2D readback waveform. In other words, the MIMO model is a thin slice of the 2D model.

Detector design for MIMO channels has been studied for more than a decade [17, 18, 19, 20, 21]. A variation of the MIMO model arises from the multi-track scenario in which the tracks are written in small groups, with guard bands between neighboring groups; this limits the number of inputs and avoids the problem of unknown boundary conditions [18, 19, 20, 21]. For example, the performance of an ideal ML detector for the multi-track scenario was analyzed in [18]. A variety of low-complexity multi-track detectors have been proposed. For example, a method in [19] divides a low density parity check (LDPC) codeword into three segments and records them on three adjacent tracks. The detector then iteratively

detects between three inner detectors and an outer decoder to recover three input tracks from $N = 3$ readback waveforms. The detector in [20] improves the recovery of the 2D-equalized center track, from $N = 3$ signals, by first estimating the sidetracks and providing their ITI information to the center track detector. A recently proposed detector [21] recovers four input tracks from $N = 2$ readback waveforms using the joint detection and decoding of two parallel detectors concatenated with two parallel LDPC decoders. Another recent detector employs MMSE linear equalization to a 1D target in order to recover the middle track from $N = 3$ readback waveforms [17].

Because of the fundamental differences between the two models, the detector design for one cannot be applied to the other. The discriminating features of the two models are:

1. In the MIMO model, unlike the 2D model, the downtrack and crosstrack dimensions are not interchangeable and the read back matrix is not a square. The downtrack dimension dominates over the crosstrack dimension with thousands of bits over a handful of readback waveforms.
2. The MIMO model can be underdetermined. The model is underdetermined in the case where there are more input tracks that significantly contribute to the readback waveforms but these tracks are not sufficiently covered by the read heads to be reliably detectable. This feature represents the unknown boundary condition which is specific to the MIMO model and is not generally a part of the 2D model.

In this thesis, we adopt the MIMO model for low-latency applications for the case when there are no guard bands, so that the number of readback waveforms is smaller than the number of contributing tracks and the system is underdetermined. We mainly focus on the synchronization component of the detector where the detector should estimate the user bits using the samples that are asynchronous to the recorded bits.

2.2 Timing Recovery/Synchronization

The previous section provided an overview of TDMR detection strategies based on a bifurcation in channel modeling. This section focuses on the timing recovery block and provides fundamental concepts to understand its functionality.

Here, we can use a basic model for 1D magnetic recording channel to explain the basics of timing recovery, since the following concepts transfer to more sophisticated models for TDMR as well.

Consider a binary pulse-amplitude modulation (PAM) signal as a model for 1D magnetic recording channel. The PAM readback signal is low-pass filtered to remove out-of-band noise, yielding:

$$s(t) = \sum_{\ell} a_{\ell} f(t - \ell T - \tau_{\ell}) + n(t), \quad (2.1)$$

where $\{a_{\ell}\} \in \{\pm 1\}$ are the uncoded information bits, $\ell T + \tau_{\ell}$ is the arrival time of the ℓ -th pulse carrying the ℓ -th bit (or the position of the ℓ -th bit in the downtrack dimension), T is the ADC sampling period and τ_{ℓ} is the delay in the arrival of the ℓ -th bit, $f(t)$ is the modulating pulse shape, assumed to be bandlimited to half the bit rate where the bit rate depends on the nature of τ_{ℓ} , and the $n(t)$ is the AWGN signal. (2.1) is a readback waveform, input to the read channel. After low-pass filtering, an ADC that samples this signal at intervals $\{kT\}$, yields:

$$s_k = \sum_{\ell} a_{\ell} f(kT - \ell T - \tau_{\ell}) + n_k \quad (2.2)$$

The problem begins when the ADC sampling times $\{kT\}$ differ from the arrival times of the pulses $\{kT + \tau_k\}$. Fig. 2.2 illustrates this situation. The solid waveform is the received waveform $r(t)$. We see that the pulses arrive at times $\{kT + \tau_k\}$, hence the ideal times for the ADC to sample are also $\{kT + \tau_k\}$. These samples are marked with empty circular markers. However, the ADC samples at times $\{kT\}$ instead, which yields *asynchronous* samples marked with a cross.

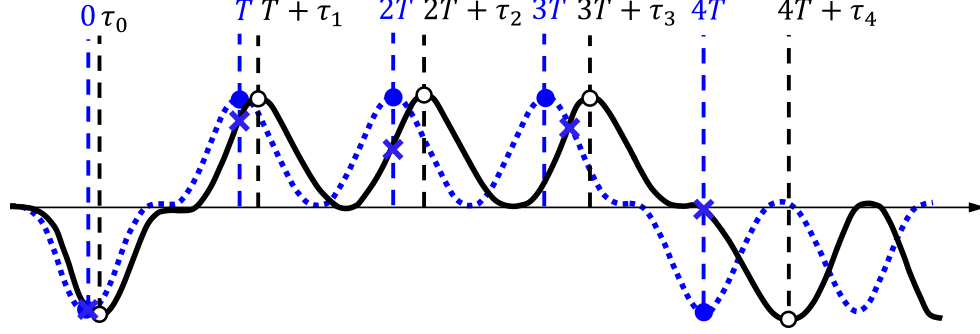


Figure 2.2: Real vs. expected readback waveform. The solid waveform is the real readback where the pulses arrive at times $\{kT + \tau_k\}$. However, the read channel expects to receive the dashed waveform where the pulses arrive at ADC sampling times $\{kT\}$.

Also, the actual arrival times $\{kT + \tau_k\}$ are unknown to the read channel, that is to say that the read channel expects the pulses to arrive at times $\{kT\}$. Therefore, the ideal read back waveform from the read channel point of view is the dashed waveform where the pulses arrive at $\{kT\}$. The difference between the actual arrivals of the pulses and the ADC sampling times is referred to as the *timing offset* τ_k . There are mainly three different models for the timing offsets, the choice of which determines which strategy should be employed for timing recovery in the read channel. The timing offsets can adopt one or a combination of different models together.

2.2.1 Models of Timing Offset

Fig. 2.3 illustrates the three main types of timing offsets in magnetic recording channel. Also, Fig. 2.4 shows the implications of the three models on data tracks, where the upper track is the track that the read channel expects, and the lower track is the real track. (The misalignment between the bits on the both tracks are unrealistically exaggerated for effect.)

The simplest case is the constant *phase* offset in Fig. 2.3 (a), where

$$\tau_k = \theta. \quad (2.3)$$

Here, the bit period is the same as the ADC sampling period T , and the delay in arrival

times $\{\tau_k\}$ is a constant θ which occurs due to an initial position offset of the read head. Therefore, using the constant phase offset model of (2.3) in the ADC outputs of (2.2), we arrive at the model for the readback waveform with a constant phase offset, according to:

$$s_k = \sum_{\ell} a_{\ell} h(kT - \ell T - \theta). \quad (2.4)$$

The expected readback waveform, according to the ADC sampling times, is the dashed waveform, and the real waveform, the solid waveform, is simply a shifted version of the expected waveform. Likewise, as Fig. 2.4 shows, the bit boundaries on the real track are shifted by a positive value of θ , to the right.

The frequency offset is widely used and one of the most important models for timing offset in magnetic recording. It results from the mismatch between the ADC and the write head clock frequencies. The frequency offsets $\{\tau_k\}$ increase or decrease linearly in time, according to:

$$\tau_k = k\Delta T, \quad (2.5)$$

where the bit period is $T + \Delta T$, and where ΔT , the *frequency offset parameter*, determines the severity of the offset. ΔT , in general, can be positive or negative with interesting implications: A positive ΔT means that the bit period is larger than the ADC sampling period, or in other words, the ADC is sampling faster than it should. Fig. 2.3 (b) shows a positive ΔT , where the actual readback waveform gradually *drifts away* from the expected waveform. Further, a positive ΔT means that the bits on the real track, Fig. 2.4 are wider than anticipated by the ADC. A negative ΔT , on the other hand, means that the bit period is smaller than the ADC sampling period, that is the ADC is sampling slower than it should. It is important to note that a negative ΔT contradicts the Nyquist sampling theorem: In the case where the underlying pulse shape is bandlimited to half the bit rate, the ADC sampling rate should be higher than the bit rate. Nevertheless, since the practical values of ΔT parameter in magnetic recording channel are quite small, even a negative frequency

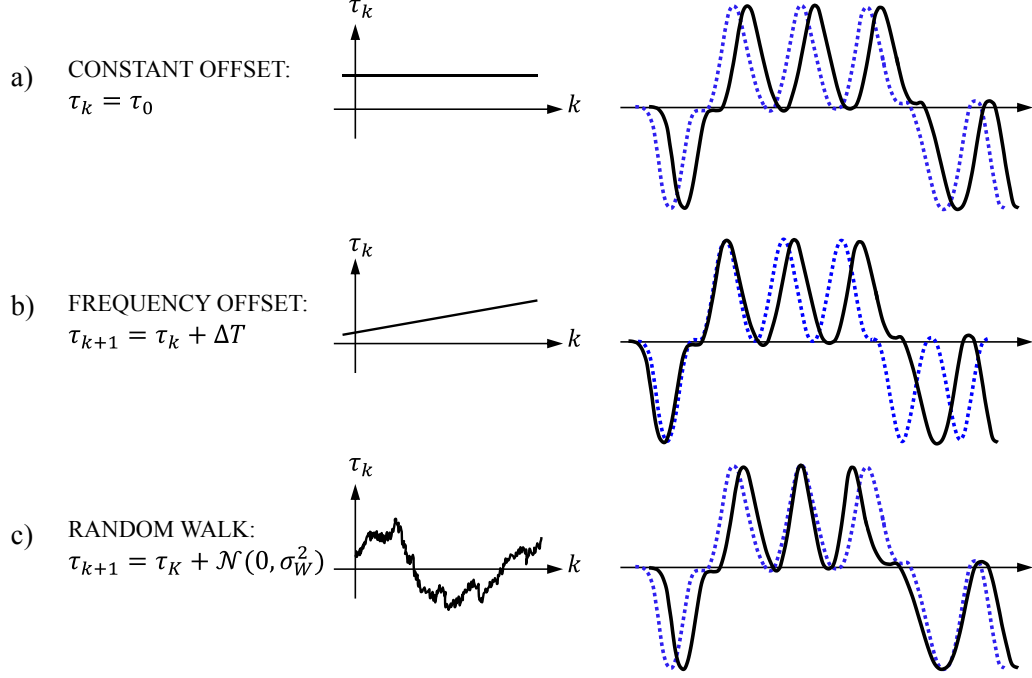


Figure 2.3: Models of timing offset in magnetic recording channel.

offset parameter does not necessary yield unreliable detection.

Using the frequency offset model of (2.5) in the ADC outputs of (2.2), we arrive at the model for the readback waveform with frequency offset:

$$s_k = \sum_{\ell} a_{\ell} h(kT - \ell T - \ell \Delta T). \quad (2.6)$$

The third model for timing offset is the random walk, as shown in Fig. 2.3 (c). Here, similar to the constant phase offset, the ADC sampling period and the bit period are the same. However, the timing offset τ_k at time k is the summation of Gaussian random variables $\{w_k\}$ with zero mean and variance σ_w^2 , up to time k , according to:

$$\tau_k = \sum_{i=0}^k w_i. \quad (2.7)$$

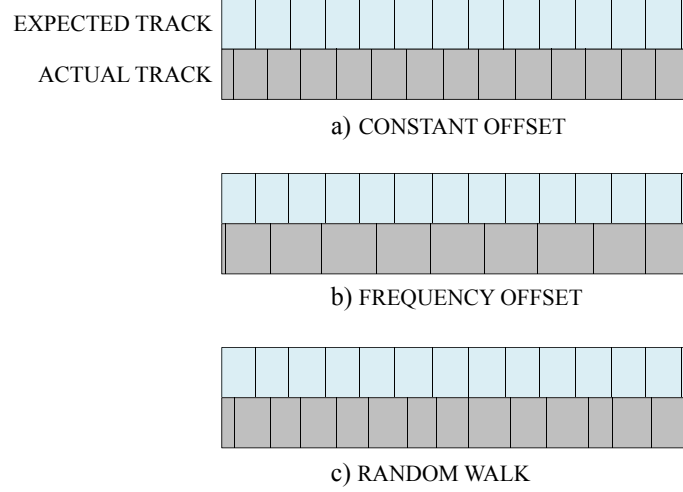


Figure 2.4: Models of timing offset in magnetic recording channel.

Therefore, the model for readback waveform with random walk offset becomes:

$$s_k = \sum_{\ell} a_{\ell} h(kT - \ell T - \sum_{i=0}^{\ell} w_i). \quad (2.8)$$

Fig. 2.4 (c) illustrates this case where the bits on the real data track have slight random misalignment compared to the bits on the expected track.

2.3 Synchronization in 1D Magnetic Recording

As mentioned, timing recovery is the process of correcting, or compensating for, the misalignment between the ADC sampling times and the actual arrival times of the bits. In this section we first layout the basics of timing recovery, next we explain the main timing recovery schemes that have been employed on 1D magnetic recording channel so far. Synchronization in 1D magnetic recording is mature. Fortunately, these strategies also apply to the current generations of TDMR channel where tracks are detected one at a time.

2.3.1 The Basics of Timing Recovery

Timing recovery works based on the principle that, if we somehow know the timing offsets $\{\tau_k\}$, we can extract the correct samples either by controlling the ADC sampling times, or

through interpolation after the ADC. To this end, the best sampling times $\{t_k\}$ are given as:

$$t_k = kT + \tau_k, \quad (2.9)$$

where T is the sampling period of a free-running ADC prior to timing recovery. Therefore, the ADC output from (2.2) becomes:

$$\begin{aligned} s_k &= \sum_{\ell} a_{\ell} f(kT + \tau_k - \ell T - \tau_{\ell}) + n_k \\ &\approx \sum_{\ell} a_{\ell} f(kT + \tau_{\ell} - \ell T - \tau_{\ell}) + n_k \\ &= \sum_{\ell} a_{\ell} f(kT - \ell T) + n_k, \end{aligned} \quad (2.10)$$

where the approximation in the second line, where τ_k is replaced by τ_{ℓ} , is valid when the timing offset varies slowly enough that it is approximately constant over the duration for which the bit response $f(t)$ is significant. Equation (2.10) is a PAM signal, sampled exactly at bit arrivals. In practice, the perfect realization of (2.10) with timing recovery is impossible because: 1) the timing offsets are unknown to the read channel and can only be estimated, and 2) since the bit response $f(t)$ is usually long, the approximation in the second line can be erroneous. Nevertheless, an estimation of (2.10) using the estimated timing offsets $\{\hat{\tau}_k\}$ does not lead to tangible performance loss either.

A basic timing recovery scheme is shown in Fig. 2.5. Here, the received signal is low-pass filtered and sampled at times $\{kT + \hat{\tau}_k\}$. Since the actual timing offsets are not known to the read channel, the loop in Fig. 2.5 is to generate estimated timing offsets $\{\hat{\tau}_k\}$ to correct the ADC sampling times.

- **Phase-Locked-Loop**

A first-order PLL updates $\hat{\tau}_k$, according to [22]:

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \alpha \hat{\epsilon}_k, \quad (2.11)$$

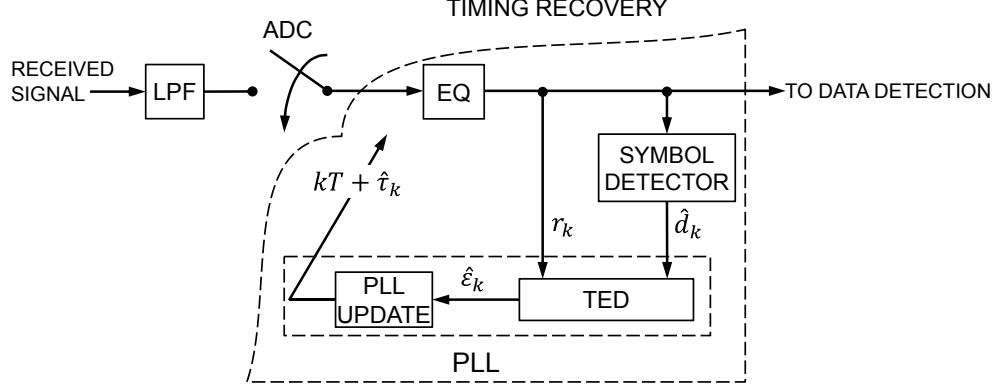


Figure 2.5: Basic timing recovery.

where $0 < \alpha < 1$ is the PLL stepsize, and $\hat{\epsilon}_k$ is an estimate of the actual timing error defined as:

$$\epsilon_k = \tau_k - \hat{\tau}_k. \quad (2.12)$$

Since τ_k is unknown, the actual timing error ϵ_k is not known either and can only be estimated. A timing error detector (TED), which will be discussed shortly, provides this estimate. The PLL operation can be intuitively explained: Assume that at time k an accurate estimate of the timing error $\hat{\epsilon}_k = \epsilon_k$ is available to the PLL. Then the PLL can simply add $\hat{\epsilon}_k = \epsilon_k$ to $\hat{\tau}_k$ ($\alpha = 1$), to get $\hat{\tau}_{k+1} = \tau_k$. If τ_k is a constant phase offset, it means that the PLL has converged to the correct timing offset at time k and therefore all $\{\hat{\epsilon}_\ell\}_{\ell \geq k}$ are zero. In practice, however, $\hat{\epsilon}_k$ is only a noisy estimate of the actual ϵ_k and therefore PLL multiplies $\hat{\epsilon}_k$ by α to attenuate the noise. The smaller the α means a better noise attenuation, but it increases the PLL rise-time and slows down PLL convergence. Therefore, α is usually adjusted based on the operating SNR region: At high SNR regions, a larger α is chosen to help PLL converge faster and also to help PLL track abrupt changes in timing offset if the timing offset is time varying. To the contrary, at low SNR regions, a smaller α should be chosen to prevent PLL divergence.

We can also study the PLL behavior when it has reached the steady-state or in-lock region. In steady-state region, the timing error estimate $\hat{\epsilon}_k$ is assumed to be linear, that

is:

$$\begin{aligned}\hat{\epsilon}_k &= \epsilon_k + \eta_k \\ &= \tau_k - \hat{\tau}_k + \eta_k\end{aligned}\tag{2.13}$$

where $\{\eta_k\}$ are i.i.d noise terms, and independent of $\{\epsilon_k\}$, and where we have used (2.12) in the second line. Therefore, (2.11) becomes a linear system, according to:

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \alpha(\tau_k + \eta_k - \hat{\tau}_k).\tag{2.14}$$

By taking the z -transform of (2.14), the transfer function of the first-order PLL is given by:

$$\frac{\hat{\tau}(z)}{\tau(z) + \eta(z)} = \frac{\alpha z}{z - 1}.\tag{2.15}$$

A perfect phase-lock is reached if the steady-state estimated timing error is zero:

$$\begin{aligned}\hat{\epsilon}_{ss} &= \lim_{k \rightarrow +\infty} \hat{\epsilon}_k \\ &= \lim_{z \rightarrow 1} (z - 1)\hat{\epsilon}(z),\end{aligned}\tag{2.16}$$

where in the second line we have applied the *final value theorem*. If the timing offset is a constant phase offset θ , where $\tau(z) = \theta z / (z - 1)$, then a perfect lock with $\hat{\epsilon}_{ss} = 0$ is achieved. We can examine this by replacing (2.13) into (2.16). However, in face of a frequency offset where $\tau(z) = \Delta T z / (z - 1)^2$, the first-order PLL exhibits a nonzero steady-state error. To achieve a zero steady-state error, a second-order PLL is used instead, where the PLL update equation is given by [22]:

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \alpha \hat{\epsilon}_k + \beta \sum_{\ell=0}^{k-1} \hat{\epsilon}_\ell.\tag{2.17}$$

Compared to the first-order PLL, we have an accumulator and an additional stepsize

β . Intuitively, if, for example, the average estimate of the timing error is positive, then the second-order PLL can correct the positive nonzero steady-state error by adding the positive third term in (2.17) to the estimated timing offset. Also, similar to the approach we used for the first-order PLL, we can examine the zero steady-state error of the second-order PLL using its transfer function, given by:

$$\frac{\hat{\tau}(z)}{\tau(z) + \eta(z)} = \frac{\alpha z + (\beta - \alpha)}{z^2 + (2 - \alpha)z + (1 - \alpha + \beta)}. \quad (2.18)$$

- **Timing Error Detector**

Consider Fig. 2.5. The TED is the most important component of a PLL. In general, a TED estimates the timing error ϵ_k of (2.13), using the ADC outputs after equalization $\{r_k\}$, and the noiseless ideal equalized samples, namely $\{d_k\}$, respectively given by:

$$r_k = \sum_{\ell} a_{\ell} h(kT + \hat{\tau}_k - \ell T - \tau_{\ell}) + n_k, \quad (2.19)$$

where $h(t)$ is the PR target, and

$$d_k = \sum_{\ell} a_{\ell} h(kT - \ell T). \quad (2.20)$$

Moreover, the TED operates in two modes: first in the *acquisition* mode and then in the *decision-directed* mode. The acquisition period refers to an initial few hundreds bits of a sector, known as the preamble, where the bits are known to the read channel. The preamble period is mainly added to a sector in order to make sure that the PLL reaches the in-lock phase before it has to track the timings of the unknown information bits. The decision-directed mode refers to the rest of the sector where the bits are unknown and should be estimated by the read channel. During the acquisition mode, the ideal equalized samples can be obtained from (2.20). During the decision-directed mode, however, the ideal equalized samples can only be estimated using a symbol

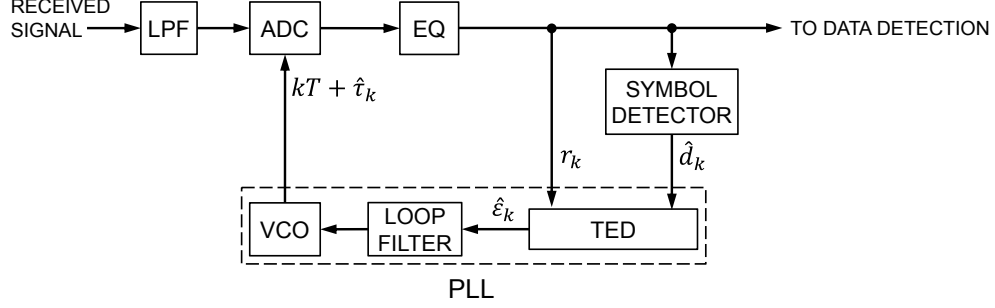


Figure 2.6: A conventional VCO-based timing recovery.

detector, according to:

$$\hat{d}_k = \sum_{\ell} \hat{a}_{\ell} h(kT - \ell T). \quad (2.21)$$

The widely-used Mueller and Müller (M&M) TED [23] computes \hat{e}_k according to:

$$\hat{e}_k = r_k \hat{d}_{k-1} - r_{k-1} \hat{d}_k. \quad (2.22)$$

Different implementations and derivatives of the timing recovery in Fig. 2.5 have evolved over time. In the followings we overview different generations of timing recovery implementations.

2.3.2 Conventional VCO-Based Timing Recovery

Initially, conventional synchronization was being performed in the analog domain with the aid of an analog voltage-controlled oscillator (VCO) in the PLL circuitry. Fig. 2.6 illustrates a conventional VCO-based timing recovery. A VCO produces an output signal whose instantaneous frequency is the signal input to the VCO. The VCO, in turn, determines the sampling times of the ADC.

Nevertheless, there were both cost and performance benefits in moving all the analog parts of Fig. 2.6 to the digital domain. All parts could be fully digitally implemented, except for the VCO. Therefore, fully-digital timing recovery architectures emerged which excluded a VCO.

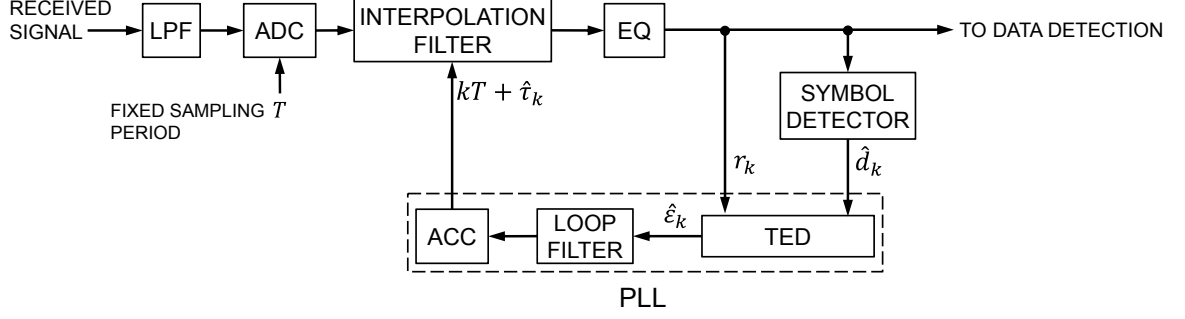


Figure 2.7: Interpolated timing recovery in 1D magnetic recording. This figure is adapted from [4] with an added equalizer after the interpolation filter (resampler).

2.3.3 Interpolative Timing Recovery

Currently, interpolative timing recovery (ITR) [4] is widely used since it is fully digital, therefore easy to implement, and since it allows the ADC to run at a free rate (above Nyquist). A block diagram of an ITR scheme in 1D magnetic recording is shown in Fig. 2.7 [4]. The core idea is that as long as the ADC is sampling above Nyquist, the ADC output samples are sufficient statistics to represent the underlying continuous-time signal. Hence, the correct samples can be recovered later, after ADC, using interpolation, and, therefore, the ADC can be left *free-running*, for example at a fixed sampling period T . As Fig. 2.7 shows, the ITR scheme is a feedback loop that extracts the synchronous samples which would have arisen if the ADC was sampling at the correct times, from the asynchronous ADC samples, as follows: At every time k , the TED [23] estimates the error between the estimated timing and the correct timing using the estimated symbol and the equalized ADC sample, according to (2.22). The estimated timing error is fed to a second-order PLL to update the new timing offset, and the new timing offset is used to recover the correct sample at time $k + 1$ via interpolation filter. After a transition time, the loop converges and locks to the correct timing offsets and will continue thereafter to track the changes in the correct timing offsets. The recovered samples can then be used by the rest of the read channel for the ultimate detection of the bits.

2.3.4 Iterative Timing Recovery

The entire read channel architecture can be formulated as a problem of jointly determining the maximum-likelihood estimates of the timing offsets and the information bits. Clearly, though, the direct implementation of such a read channel is prohibitively complex, hence all practical strategies are only approximates of the ideal read channel. In the *classical* timing recovery, explained so far and depicted in Fig. 2.8 (a), the timing offsets and the bits are estimated separately and sequentially. Since the classical timing recovery performs timing recovery *prior* to and *separate* from the equalization and decoding parts, it ignores the presence of the code, and assumes that the bits are mutually independent. Therefore, as expected, the classical timing recovery falls short in approximating the ideal read channel and can fail in low SNR regions, the same regions where the equalization and decoding succeed because of the powerful codes used in magnetic recording. In order to take advantage of the presence of powerful codes, an alternative approach was proposed in [24], where timing recovery is added to each iteration of a turbo equalizer that performs equalization and decoding iteratively in a loop, as depicted in Fig. 2.8 (b). The advantage of *iterative* timing recovery is twofold: 1) at each global iteration, timing recovery provides a better estimate of the timing offset using the new estimates of the decoded bits, and 2) since the timing updates is added to the already-existing loop, the added complexity, compared to a classical read channel that performs separate timing recovery and turbo equalization, is minimal.

All timing recovery schemes of Fig. 2.6, Fig. 2.7, and Fig. 2.8 use a TED that uses the estimated symbol and the ADC output at time k to provide an estimate of the timing error at time k . Nevertheless, the symbol detector which provides one of the inputs to the TED suffers from an inherent trade-off between the reliability and the delay in the detected symbols: Consider a Viterbi detector, for example. At every time k , Viterbi provides estimated bits from time 0 up to time k . Inherently, in the Viterbi algorithm there will be a merge between different survivor paths from time 0 to time $k - D$, for example. This means that the estimated bits from time $k - D + 1$ to time k are not as reliable as those

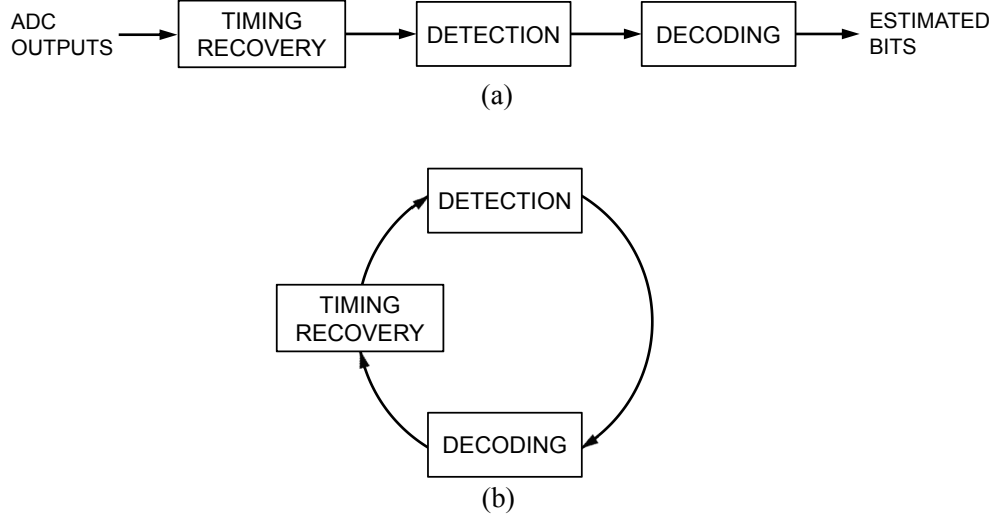


Figure 2.8: (a) Read channel with classical timing recovery, vs. (b) iterative timing recovery.

from time 0 to time $k - D$. Therefore, a bigger delay in the detected symbols translates to more reliable symbols that boosts the TED to produce more accurate estimates of the timing errors which in turn result in more accurate estimates of the timing offsets. On the other hand, however, a delayed estimated symbol delays the estimated timing offset and therein prevents the entire timing recovery to be able to track fast changes in the timings of the received waveform. This trade-off is a draw-back of the timing recovery schemes mentioned so far. In the following, we explain a novel timing recovery scheme that not only enjoys a *zero* decision delay in estimating the timing offsets but also is a closer implementation of the ideal read channel.

2.3.5 Per-Survivor Processing (PSP) for Timing Recovery

To overcome the reliability-versus-delay drawback inherent in all timing recovery schemes mentioned above, a reliable decision with zero delay can be extracted by utilizing the already-given information in the trellis structure of a trellis-based detector. The idea of using the information available in the trellis to estimate unknown parameters is known as *Per-Survivor Processing* (PSP) [25]. The gist of PSP is that each branch in the trellis uniquely corresponds to a specific decision. Then, at least one branch at every stage corresponds to the correct

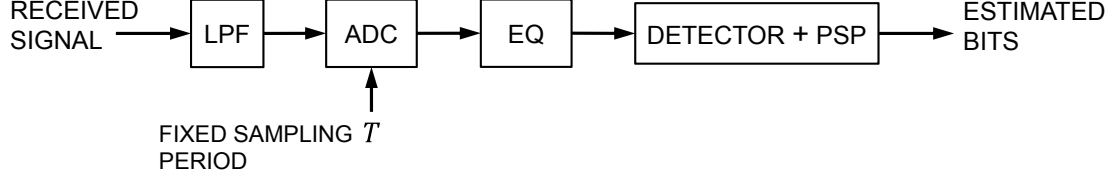


Figure 2.9: The read channel with per-survivor processing for timing recovery in 1D magnetic recording.

decision. Therefore, by utilizing the correct decision in updating the estimated timing offset, the decision delay will inherently be zero. More importantly, since this scheme embeds synchronization *inside* the trellis-based detection, and thereby jointly performs timing recovery and detection of the bits, theoretically, it is *the closest* implementation of the ideal ML read channel that is available by far.

The idea of using PSP for timing recovery in magnetic recording was proposed in [26] and developed in [27], initially to work within the Viterbi detector in order to detect uncoded bits. Later, PSP for timing recovery was developed in [28] to also work within BCJR detector in order to detect coded bits. PSP has been employed in many other applications, including channel identification and adaptive ML sequence detection in [25], and phase/carrier recovery in [29].

Fig. 2.9 illustrates the overall read channel where the timing recovery and detection are performed jointly using a Viterbi or BCJR detector with an embedded PSP algorithm. Since PSP for timing recovery embedded inside a Viterbi detector is widely used in this thesis, in the following, we provide an explanation to the algorithm for detecting uncoded bits. Consider the read channel in Fig. 2.9. The ADC is sampling asynchronously to the bit rate and with a fixed sampling period T . For clarity of expression we assume that the equalization to a PR target is perfect. Therefore, the inputs to the detector embedding the PSP algorithm are the ADC asynchronous samples after the equalization, and can be written as:

$$r_k = \sum_{\ell} a_{\ell} h(kT - \ell T - \ell \Delta T) + n_k, \quad (2.23)$$

where we have assumed a frequency offset model for $\tau_k = k\Delta T$, according to (2.6). To aid with the explanation, we further assume an example where the target $h(t) = g(t) - g(t - 2T)$ is the PR-IV pulse shape where $g(t) = \sin(\pi t/T)/(\pi t/T)$.

A conventional Viterbi algorithm would be an optimal ML detector to detect the bits if the frequency offset parameter $\Delta T = 0$. With $\Delta T \neq 0$, however, a PSP-based timing recovery adds additional timing update operations to the standard Viterbi to estimate the correct samples for the optimal detection. In particular, since each branch in the trellis structure is associated with a specific bit, each branch gives a different estimate of the timing offset, where at least one of the branches with its estimate of the timing offset is correct. The key idea, therefore, is to resample the analog readback waveform using different timing offsets associated with different branches. As the Viterbi algorithm progress through the trellis and updates the survivor path for each state, the PSP, in parallel, updates the timing offset estimate for each survivor pass. This means there is one PLL for every survivor path in the trellis.

Algorithm 1 describes the pseudocode of the Viterbi detector with PSP-based timing recovery. The lines marked with an asterisk show the additional steps due to the PSP and beyond the standard Viterbi algorithm. The input to the Algorithm is the equalized ADC outputs of (2.23), and the output is the estimated information bits. The algorithm begins by setting the initial state to state 0 in line 1. In line 2, an empty vector $S(p)$ for each survivor path for every states p is declared. Line 3 initiates an estimate of the timing offset for every state p . (We assumed a frequency offset model with zero initial phase offset.) Also, line 4 initiates a variable *sum* for every state p . This variable will be used later in the algorithm to accumulate the past estimates of the timing error for each state in the trellis. The loop from line 5 to line 15 steps into each stage of the trellis.

Consider the algorithm runs on a PR-IV trellis as shown in Fig. 2.10. Let $\{a_{k-2}a_{k-1}\}$ denote the state at time k , or stage k , in the trellis. Since the PR-IV response has 2 memory taps, there is a total of $Q = 2^\mu = 4$ states in the trellis. Also, let (p, q) denote the state

transition from state p to state q . Since the alphabet is binary, there are two incoming branches to every state q . For example, two transitions $(2, 3)$ and $(3, 3)$ arrive at state 3 at time $k + 1$. In order to select the best transition, the algorithm first resamples the readback waveform using the $\hat{\tau}_k(2)$ and $\hat{\tau}_k(3)$, in line 7. Then, in line 8, the two branch metrics $\gamma(2, 3)$ and $\gamma(3, 3)$ are computed using the ideal ADC outputs corresponding to the two transitions. The transition which leads to the minimum partial path metric $\pi_{k+1}(3)$ is selected according to the Viterbi, in line 9 where $\Phi_k(p)$ denotes the partial path metric of state p at time k . Next in line 10, $\Phi_k + 1(3)$ is updated with the path metric of the selected transition. Also, the survivor path of state 3 $\Phi_k + 1(3)$ is extended in line 11 to include the selected transition. From line 12 to line 14, the algorithm updates the timing offsets of state 3 to be used in the next stage. Line 12 implements the M&M TED [23], according to (2.22), for the ending state 3 and using the information on the updated survivor path of state 3. Line 13 and line 14 compute the updated timing offset estimate of state 3 using a second-order PLL with stepsizes α and β . The same operations are performed for the entire length of the trellis. Finally, in line 17, the estimated information bits are extracted from the path with the minimum path metric.

Since PSP-based timing recovery needs to run one PLL for each survivor path in the trellis, the computational complexity is Q times more than classical timing recoveries in Fig. 2.8 (a), Fig. 2.7, and Fig. 2.6, where timing recovery is performed separately and prior to the detection.

2.4 Synchronization for TDMR

It is clear from the previous section that synchronization in 1D magnetic recording is well established. In this section we transition to the problem of timing recovery where significant degrees of ITI prevents us to consider data tracks in isolation. Specifically, the problem is to detect data tracks that exhibit different timing offsets, using all available readback waveforms. Unlike 1D magnetic recording, prior works that address timing recovery specific

Algorithm 1: Viterbi with PSP

Inputs: equalized ADC outputs $\{r_k\}$
Output: $\hat{\mathbf{a}}$

- 1 **Init:** $\Phi_0(0) = 0, \Phi_0(p) = \infty \forall p \neq 0$
- 2 **Init:** $\mathbf{S}_0(p) = [\] \forall p$
- *3 **Init:** $\hat{\tau}_0(p) = 0 \forall p$
- *4 **Init:** $sum(p) = 0 \forall p$
- 5 **for** $k = 0$ **to** $L + \mu - 1$ **do**
- 6 **for** $q = 0$ **to** $Q - 1$ **do**
- *7 $r_k(p) = r(kT + \hat{\tau}_k(p)) \forall p \rightarrow q$
- 8 $\gamma_k(p, q) = |\hat{r}_k - \hat{d}_k(p, q)|^2 \forall p \rightarrow q$
- 9 $\pi_{k+1}(q) = \underset{p}{\operatorname{argmin}}\{\Phi_k(p) + \gamma_k(p, q)\}$
- 10 $\Phi_{k+1}(q) = \Phi_k(\pi_{k+1}(q)) + \gamma_k(\pi_{k+1}(q), q)$
- 11 $\mathbf{S}_{k+1}(q) = [\mathbf{S}_k(\pi_{k+1}(q)) | \pi_{k+1}(q)]$
- *12 $\hat{\epsilon}_k(q) = r_k(\Phi_{k+1}(q))\hat{d}_{k-1}(\Phi_k(\Phi_{k+1}(q)), \Phi_{k+1}(q))$
 $-r_{k-1}(\Phi_k(\Phi_{k+1}(q)))\hat{d}_k(\Phi_{k+1}(q), q)$
- *13 $sum(q) = sum(\pi_{k+1}(q)) + \hat{\epsilon}_{k-1}(\pi_{k+1}(q))$
- *14 $\hat{\tau}_{k+1}(q) = \hat{\tau}_k(\pi_{k+1}(q)) + \alpha\hat{\epsilon}_k(q) + \beta sum(q)$
- 15 **end**
- 16 **end**
- 17 Extract $\{\mathbf{a}\}$ from the survivor path that minimizes $\Phi_{L+\mu}$

to the TDMR channel, as we overview them in the following, are rare.

Here, timing recovery architecture, similar to every other part of the read channel, is heavily influenced by the model chosen for the TDMR channel. Therefore, we overview timing recovery for TDMR following the bifurcation in TDMR channel modeling of Sect. 2.1.

2.4.1 2D ISI Model

In case a large matrix of observations, for example 1000-by1000 bits, is available for processing, the 2D ISI model is suitable. In order to perform PLL-based timing recovery on a 2D setting, a 2D PLL including a 2D TED is required. A 2D extension of the M&M TED [23] has been proposed in [30]. Their approach considers a separable model for the

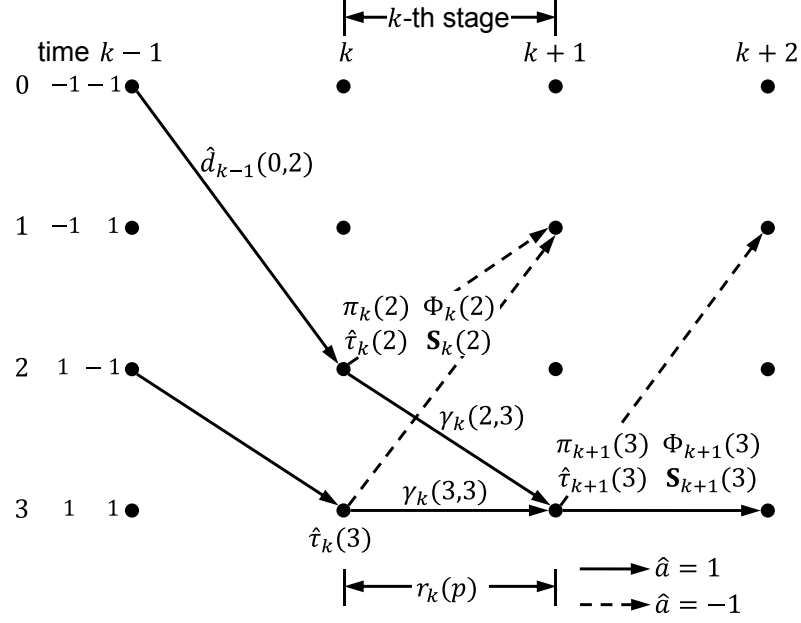


Figure 2.10: Viterbi algorithm with PSP-based timing recovery on PR-IV trellis.

timing offsets, that is the timing offsets in the downtrack and crosstrack dimensions are independent of one another which is not necessarily the case for TDMR. Because, if the read head has a position offset (phase offsets in both dimensions), then a frequency offset in one dimension causes a frequency offset in the other dimension as well.

A 2D PLL for updating a non-separable timing offset for TDMR is recently proposed in [31]. Here, a 2D TED computes the angle between a vector of asynchronous ADC samples on a 2D grid and the corresponding vector of ideal samples. Also, the 2D PLL update equations are the exact, straightforward 2D extensions of the second-order 1D PLL. The 2D PLL stability criteria, noise performance, and the loop bandwidth, however, are remaining to be addressed. In a following work [32], a 2D interpolation filter is derived, in order to work within an ITR scheme including the 2D PLL derived earlier. This 2D interpolation filter is the 2D extension of the 1D interpolation filter proposed in [33] that was computed through minimizing the mean-squared error between the ADC samples and the ideal samples.

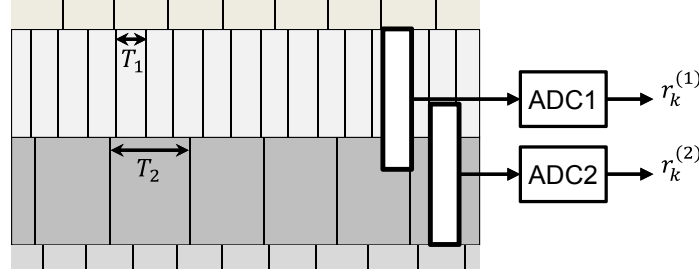


Figure 2.11: A MIMO model with $K = 2$ asynchronous tracks and $N = 2$ overlapping read heads. (The asynchronous between bit rates are exaggerated for effect.)

2.4.2 MIMO Model

The delay caused to provide a large scanning of the disk is a prohibitive factor that affects the applicability of the state-of-the-art works in the 2D ISI setting for TDMR application where we cannot afford to wait for hundreds of revolutions of the disk to accumulate large scans. Therefore, for this thesis, we were prompted to focus on synchronization and detection based on the MIMO model for low latency applications. The objective is to recover as many bits as possible from a single scan of multiple readers.

The problem is to detect one or more tracks, from one or more readback waveforms including considerable ITI, when the tracks to be detected were written asynchronously, meaning that neither the bit rates (frequency) nor the bit boundaries (phase) are aligned between neighboring tracks. An example of a MIMO model with $K = 2$ asynchronous track and $N = 2$ readers with significant overlap are shown in Fig. 2.11. As shown, the bit rate of track 2 is much smaller than the bit rate of track 1. (This difference is exaggerated for effect.) The problem is to detect the two tracks using the two readback waveforms. There are two approaches to this problem:

- 1) **Single-Track Detection:** The first approach is to use the two waveforms to recover the bits from each track, separately. This approach is the approach taken by current implementations of TDMR read channels in data storage industry [34]. Read channel designs according to this approach should follow an initial step where the ITI is mitigated and/or canceled as much as possible prior to the detection of a single track

of interest. The immediate advantage is that from some point forward in the read channel, the problem is reduced to a 1D detection problem which can exploit the well-established 1D synchronization and detection strategies. All the timing recovery schemes in 1D magnetic recording of Sect. 2.3 in conjunction with TDMR detection strategies of Sect. 2.1 form the prior works following this approach.

Therefore, in the first part of this thesis, in chapters 3 and 4, we used multiple readers to recover the bits from a single track of interest at a time. Here, our goal was to improve the synchronization and detection strategies employed by current generations of TDMR read channels. In particular, we have studied:

- (a) the problem of mitigating ITI in detecting on a track-by-track basis when all the contributing tracks are synchronous, i.e. they have the same bit rate and bit boundaries (chapter 3).
- (b) the problem of timing recovery and mitigating ITI for detecting on a track-by-track basis when all the contributing tracks have different bit rates and bit boundaries (chapter 4).
- (c) And also, the problem of PR equalization and timing recovery for detecting on a track-by-track basis when all the contributing tracks have different bit rates and bit boundaries (chapter 4).

2) Multitrack Detection: In the second part of this thesis, we study the problem of jointly detecting multiple asynchronous tracks. Consider the problem of Fig. 2.11. An *ideal* read channel would jointly estimate the timings and the bits on the two tracks. Theoretically, the ideal read channel performs ML estimation of the timings and the bits on the two tracks all together. A joint Viterbi detector, for example, is an efficient implementation of the optimal ML detector for joint detection of only *synchronous* tracks. For detecting *asynchronous* tracks, however, the problem fundamentally changes: All known implementations of the ML detector, for example the Viterbi

detector, assume that the two readback waveforms are somehow synchronized to both of the contributing tracks. *A closer look at Fig. 2.11, however, reveals that this is impossible to achieve. Even when the exact correct timings are known, synchronizing the two waveforms to the timings of track 1, for example, necessarily desynchronizes them to the timings of track 2, and vice versa.*

To our knowledge, there is *no prior published work* that addresses the joint detection of multiple asynchronous tracks. Therefore, to realize the full potential of TDMR [5], in chapter 5, we propose the *ROTAR* algorithm for joint detection of multiple asynchronous tracks. ROTAR algorithm implements a joint Viterbi based on a time-varying partial response that results when the asynchrony of the tracks are absorbed into the underlying partial response. ROTAR also uses PSP to estimate the unknown timings of the tracks being detected.

2.5 Summary

In this chapter, we provided an overview on TDMR detection strategies based on the channel model chosen. The type of channel model divides all prior works into two distinct categories of 2D ISI model and MIMO model. Those strategies which assume a 2D ISI model require a large scan of the disk prior to the processing and thereby introduce a significant delay. In this thesis we follow the MIMO modeling approach that considers the detection problem of a few number of tracks from one or more readback waveforms, for low-latency applications.

Next, we provided an overview of the basics of timing recovery in 1D magnetic recording where efficient strategies have been proposed for decades. For TDMR channel, on the other hand, all prior synchronization work fall into the same categorization based on channel modeling. The prior work based on 2D ISI model is extremely rare, while following the MIMO approach and single-track detection, the well-established synchronization and detection strategies for 1D magnetic recording can be employed. This is the state-of-the-art read channel architectures that are currently being implemented in the industry.

In this thesis, we study the detection and synchronization problem for TDMA channel within two categories:

1. Synchronization for single-track detection: We study this problem in Chapters 4 and 5. In Chapter 3, we propose a soft ITI cancellation strategy for detecting several synchronous tracks from several readback waveforms. In the first part of Chapter 4, we extend this strategy to include synchronization of asynchronous tracks. In the second part of Chapter 4, we study the PR equalization of one or more read back waveforms when the contributing tracks have different phase and frequency offsets. Here, we propose to switch the conventional order in which synchronization is performed *prior* to equalization. We propose to significantly reduce the computational complexity by synchronizing *after* equalization.
2. Synchronization for multitrack detection: Synchronization for joint detection of multiple tracks has no prior published solution. In Chapter 5, we propose a ROTAR algorithm that is based on a time-varying partial response that results when the asynchrony of the tracks are absorbed into the underlying partial response. ROTAR estimates the timings of the tracks being detected using PSP for timing recovery.

CHAPTER 3

SOFT INTERTRACK INTERFERENCE CANCELLATION STRATEGY [35]

In this chapter we explain our initial work in detecting multiple tracks one-by-one, using multiple readback waveforms from a MIMO channel for TDMR. Here, the main objective is to cancel and/or mitigate ITI in the presence of prominent media noise. In this chapter only, there is no synchronization challenge since all contributing tracks are synchronous to one another and to the ADC's sampling rate $1/T$, which is the same as the bit rate.

3.1 Channel Model

In this chapter, we consider a linear and separable model for the channel, including first-order jitter noise and electronic noise. Later in Chapter 4, Sect. 4.2, we drop this assumption and continue as such throughout the thesis. Hence, the readback waveform for the i -th read head is given by

$$r_i(t) = \sum_j g_{i,j} \left(\sum_{\ell} a_{\ell}^{(j)} h(t - \ell T) + \left(a_l^{(j)} - a_{\ell-1}^{(j)} \right) j_l^{(j)} q(t - \ell T) \right) + n_i(t), \quad (3.1)$$

where $g_{i,j}$ is crosstrack response gain from track j at read head i , $a_k^{(j)} \in \{\pm 1\}$ is the k -th coded bit of track j , $h(t)$ is the common bit response for all tracks, T is the bit period, $j_k^{(j)}$ is the k -th jitter noise component for track j , $q(t)$ is the derivative of the corresponding transition response, and $n_i(t)$ is the additive electronic noise for the i -th read head. We also assume that the jitter components $\{j_k^{(n)}\}$ are independent identically distributed zero-mean Gaussian random variables with a variance σ_j^2 , independent from the track index j . We assume that the electronic noise $n_i(t)$ is white and Gaussian with a power-spectral density of $N_0/2$ which is the same for all read heads. The waveform in (3.1) is filtered by an

antialiasing filter and sampled at bit rate yielding

$$r_k^{(i)} = \sum_j g_{i,j} \left(x_k^{(j)} + m_k^{(j)} \right) + n_k^{(i)}, \quad (3.2)$$

where $\{x_k^{(j)} = \sum_\ell a_\ell^{(j)} h_{k-\ell}\}$ is the “ISI symbol” sequence for track j , h_k is the k -th sample of the filtered bit response, $\{m_k^{(j)} = \sum_\ell (a_\ell^{(j)} - a_{\ell-1}^{(j)}) j_\ell^{(j)} q_{k-\ell}\}$ is the data-dependent “media-noise” sequence for track j , q_k is the k -th sample of the filtered derivative of the transition response, and $n_k^{(i)}$ is the k -th sample of the filtered electronic noise $n_i(t)$, with zero mean and variance $N_0/(2T)$.

By vectorizing equation (3.2) over N read heads at time k , we arrive at a MIMO model for the TDMR channel

$$\mathbf{r}_k = \sum_j \left(x_k^{(j)} + m_k^{(j)} \right) \mathbf{g}_j + \mathbf{n}_k, \quad (3.3)$$

where $\mathbf{r}_k = [r_k^{(1)}, \dots, r_k^{(N)}]^T$, $\mathbf{n}_k = [n_k^{(1)}, n_k^{(2)}, \dots, n_k^{(N)}]^T$, and $\mathbf{g}_j = [g_{1,j}, g_{2,j}, \dots, g_{N,j}]^T$. In this MIMO model, the number of outputs is N , the number of read heads, and the number of inputs K is the number of relevant tracks that contribute to the output vector \mathbf{r}_k . The number K depends on the extent of the crosstrack response $\{g_{i,j}\}$. For example, if $\{g_{i,j}\} = 0$ for $j > K$, then the number of inputs (contributing tracks) is K .

3.2 Two Detection Strategies

We explain two strategies for mitigating ITI: 1) linear combining, and 2) soft intertrack interference cancellation strategies. The first strategy is a component of the second strategy and also will be used as a basis of comparison for the second strategy.

3.2.1 Linear Combining

In detecting every track t , we can suppress ITI by taking a linear memoryless combination of N readback waveforms according to

$$y_k^{(t)} = \mathbf{w}_t^T \mathbf{r}_k. \quad (3.4)$$

where \mathbf{w}_t is a vector of combining weights to detect track t . The weights \mathbf{w}_t should be selected to minimize the ITI and not the ISI or the media noise. Therefore, a good optimization criteria for computing these weights is the mean-squared error between the combination output and the noiseless ISI plus media noise symbols, according to

$$MSE = E \left(\left(y_k^{(t)} - \left(x_k^{(t)} + m_k^{(t)} \right) \right)^2 \right). \quad (3.5)$$

We show in Appendix A that the weights minimizing the mean-squared error are

$$\mathbf{w}_t = \left(\sum_j \mathbf{g}_j \mathbf{g}_j^T + \left(\frac{N_0}{2E_h + TM_0} \right) \mathbf{I} \right)^{-1} \mathbf{g}_t, \quad (3.6)$$

where $E_h = E_x T$ and $E_x = E((x_k^{(t)})^2)$, respectively, are the energy of the bit response $h(t)$, and variance of the ISI symbols, and where $M_0 = 4\sigma_j^2 E_q$ is the equivalent single-sided rectangular power spectrum of the media noise, expressed in terms of $E_q = \sum_k q_k^2$. If we substitute (3.3) into (3.4), we find

$$y_k^{(t)} = x_k^{(t)} \mathbf{w}_t^T \mathbf{g}_t + \sum_{j \neq t} x_k^{(j)} \mathbf{w}_t^T \mathbf{g}_j + \sum_j m_k^{(j)} \mathbf{w}_t^T \mathbf{g}_j + \mathbf{w}_t^T \mathbf{n}_k, \quad (3.7)$$

where the first term is the desirable ISI symbol of track t , biased by a factor of $\mathbf{w}_t^T \mathbf{g}_t$. We remove this bias to arrive at a conventional 1D recording model as follow

$$z_k^{(t)} = x_k^{(t)} + \eta_k^{(t)}, \quad (3.8)$$

where $z_k^{(t)} = y_k^{(t)} / (\mathbf{w}_t^T \mathbf{g}_t)$, and $\eta_k^{(t)} = (\sum_{j \neq t} x_k^{(j)} \mathbf{w}_t^T \mathbf{g}_j + \sum_j m_k^{(j)} \mathbf{w}_t^T \mathbf{g}_j + \mathbf{w}_t^T \mathbf{n}_k) / (\mathbf{w}_t^T \mathbf{g}_t)$ is the sum of the residual interference, media, and electronic noise, whose variance is

$$\sigma_{\eta}^{2(i)} = \frac{1}{2T} \times \frac{\left(2E_h \sum_{n \neq i} (\mathbf{w}_i^T \mathbf{g}_n)^2 + M_0 \sum_n (\mathbf{w}_i^T \mathbf{g}_n)^2 + N_0 \|\mathbf{w}_i\|^2 \right)}{(\mathbf{w}_i^T \mathbf{g}_i)^2}. \quad (3.9)$$

The model in (3.8) looks like a conventional 1D recording model, with an ISI sequence from track t corrupted by media and additive noise, and it can be detected using any of a variety of standard techniques, including the Viterbi detector, the BCJR detector, a pattern-dependent noise-predictive detector [4], or an iterative detector, such as a turbo equalizer that iterates between a channel detector and an error-control decoder [36] (Section 3.2.3).

The same linear ITI suppression strategy may be applied separately for each individual track to detect them one-by-one.

3.2.2 Soft ITI Cancellation

We apply the idea of successive interference cancellation that was originally developed for CDMA applications [37, 38], to TDMR. We also replace hard decisions by soft decisions to improve detection performance [39, 40, 41, 42]. In particular, we propose to detect tracks one by one and cancel ITI from previously detected tracks while accounting for the reliability of the previous decisions. Here, we detect tracks according to an ordered list Π of track indices.

The *detection order* is an important degree of freedom for the proposed detector, since the order in which tracks are detected will significantly impact performance. We will represent the detection order by an ordered list Π of track indices, where the first entry of the list is the index of the track detected first, the second entry is the index of the track detected second, and so on.

There may be an advantage in detecting a particular track more than once, and thus we allow for repeated entries in Π . For example, $\Pi = [1, 2, 3, 2, 1]$ would mean that we first

detect track 1, then track 2, then track 3, then we redetect track 2, and finally we redetect track 1. We note that a repeated detection for a given track is an option, not a necessity, and further that when it does occur it is based on the original set of readback waveforms, not a new set based on a rescan of the disk. In other words, the entire algorithm operates on a single set of sampled waveforms from a single pass of the readers over the track(s) of interest. Although performance can be improved when a second set of waveforms is made available through a repeated pass of the readers, at the cost of increased delay, this paper does not consider such extensions.

Suppose we are currently detecting track t and let \mathcal{P} denote the set of previously detected tracks, excluding the current track t . To detect track t , We *softly* cancel the interference caused by the set \mathcal{P} and linearly suppress any interference that remains. In particular, we propose to first subtract a soft estimate of the interference from the previously detected tracks in \mathcal{P} before taking a linear combination, according to

$$y_k^{(t)} = \mathbf{w}_t^T \left(\mathbf{r}_k - \sum_{j \in \mathcal{P}} \tilde{x}_k^{(j)} \mathbf{g}_j \right). \quad (3.10)$$

This equation captures the essence of our proposed soft ITI cancellation strategy. Here, $\tilde{x}_k^{(j)}$ denotes a soft estimate of the k -th ISI symbol $x_k^{(j)}$ from the previously detected track j , which is computed by convolving a sequence of soft estimates $\{\tilde{a}_k^{(j)}\}$ of the bits with the ISI response, according to the following lemma:

Lemma 1. *Let $\lambda_k^{(j)} = \ln(P(a_k^{(j)} = 1|\{\mathbf{r}_i\})/P(a_k^{(j)} = -1|\{\mathbf{r}_i\}))$ denote the k th a posteriori log-likelihood ratio (LLR) for track j . Given knowledge of $\{\lambda_k^{(j)} : \text{for all } k\}$, the soft estimate $\tilde{x}_k^{(j)}$ that minimizes the mean-squared error $E((\tilde{x}_k^{(j)} - x_k^{(j)})^2|\{\lambda_k^{(j)}\})$ is*

$$\tilde{x}_k^{(j)} = \sum_{\ell} \tilde{a}_{\ell}^{(j)} h_{k-\ell}, \quad (3.11)$$

where $\tilde{a}_k^{(j)} = \tanh(\lambda_k^{(j)}/2)$.

Proof. Setting to zero the partial derivative of $E((\tilde{x}_k^{(j)} - x_k^{(j)})^2 | \{\lambda_k^{(j)}\})$ with respect to $\tilde{x}_k^{(j)}$ leads to the result that the estimate that minimizes the mean-squared error is the conditional mean

$$\tilde{x}_k^{(j)} = E \left(x_k^{(j)} | \{\lambda_k^{(j)}\} \right) \quad (3.12)$$

$$= E \left(\sum_{\ell} a_{\ell}^{(j)} h_{k-\ell} | \{\lambda_k^{(j)}\} \right) \quad (3.13)$$

$$= \sum_{\ell} E \left(a_{\ell}^{(j)} | \{\lambda_k^{(j)}\} \right) h_{k-\ell} \quad (3.14)$$

$$= \sum_{\ell} \tilde{a}_{\ell}^{(j)} h_{k-\ell} \quad (3.15)$$

where we have introduced the conditional mean $\tilde{a}_k^{(j)} = E(a_k^{(j)} | \{\lambda_k^{(j)}\})$, which is well-known to reduce to $\tilde{a}_k^{(j)} = \tanh(\lambda_k^{(j)}/2)$ [43]. \square

Here, two extreme cases are noteworthy. In the extreme case when a previously detected track n has an infinite SNR, the resulting soft estimates $\tilde{x}_k^{(n)}$ will exactly match the actual $x_k^{(n)}$. In this case, the soft cancellation in (3.10) reduces to hard cancellation, and it will completely remove the influence of the ISI symbols of track n . At the other extreme, if track n has a zero SNR, the resulting LLR and soft decisions will also be zero. In this case, the soft cancellation in (3.10) will subtract zero, which means it will not do any cancellation at all. In practice, of course, the SNR will be between the two extremes, so that in practice the cancellation will be only partial—residual ITI will remain even after the soft cancellation process.

In (3.10), we see that, after the soft cancellation of ITI from previously detected tracks, the combining weights w_i are used to linearly suppress any ITI that remains. This includes not only ITI from as-yet undetected tracks, but also residual ITI that remains after the soft cancellation process from previously detected tracks.

In Appendix B, we show that the linear combining weights that minimize the MSE after

soft cancellation of the previously detected tracks are

$$\mathbf{w}_t = \left(\sum_j \left(\frac{\alpha_j 2E_h + TM_0}{2E_h + TM_0} \right) \mathbf{g}_j \mathbf{g}_j^T + \left(\frac{N_0}{2E_h + TM_0} \right) \mathbf{I} \right)^{-1} \mathbf{g}_t \quad (3.16)$$

where

$$\alpha_j = \begin{cases} 1, & \text{for } j \notin \mathcal{P} \\ E((x_k^{(j)} - \tilde{x}_k^{(j)})^2)/E_x, & \text{for } j \in \mathcal{P} \end{cases}. \quad (3.17)$$

For $j \in \mathcal{P}$, we can interpret α_j as a reliability factor, since it is a number between 0 and 1 that quantifies the reliability of the decisions from track j . Two extreme cases lend insight as follows:

1. At one extreme, $\alpha_j = 1$ corresponds to the case where the decisions of track j are completely unreliable; in this case, the weight computation in (3.16) will treat track j as an undetected track. Observe that for the special case when $\alpha_j = 1$ for all tracks (which happens when no tracks have been previously detected, so that \mathcal{P} is the empty set), the weights in (3.16) reduce to the linear weights of (3.6).
2. At the other extreme, $\alpha_j = 0$ corresponds to the case where track j produces completely reliable decisions, in which case the cancellation in (3.10) of the ISI symbols from track j is perfect; nevertheless, there will always be media noise from track j that is not canceled by (3.10), and for this reason the contribution from track j to the weights in (3.16) is small when $\alpha_j = 0$, but not zero.

Note that, when detecting the very first track, there will be no previously detected tracks, so that \mathcal{P} is empty. In this case, (3.10) reduces to the linear detector, and the weights of 3.16 reduce to the linear weights from (3.6). Thus, it follows that, in the proposed soft ITI cancellation scheme, the first track is detected linearly.

Similar to the linear combining case, substituting (3.3) into (3.10) yields

$$y_k^{(t)} = \mathbf{w}_t^T \mathbf{g}_t x_k^{(t)} + \mathbf{w}_t^T \times \left(\sum_{j \neq t} x_k^{(j)} \mathbf{g}_j + \sum_j m_k^{(j)} \mathbf{g}_j + \mathbf{n}_k - \sum_{j \in \mathcal{P}} \tilde{x}_k^{(j)} \mathbf{g}_j \right), \quad (3.18)$$

We remove the bias $\mathbf{w}_t^T \mathbf{g}_t$, to arrive at the conventional 1D recording model as follow:

$$z_k^{(t)} = x_k^{(t)} + \eta_k^{(t)} \quad (3.19)$$

where $z_k^{(t)} = y_k^{(t)} / (\mathbf{w}_t^T \mathbf{g}_t)$, and $\eta_k^{(t)} = \mathbf{w}_t^T \left(\sum_{j \neq t, j \notin \mathcal{P}} x_k^{(j)} \mathbf{g}_j + \sum_{j \in \mathcal{P}} (x_k^{(j)} - \tilde{x}_k^{(j)}) \mathbf{g}_j + \sum_j m_k^{(j)} \mathbf{g}_j + \mathbf{n}_k \right) / (\mathbf{w}_t^T \mathbf{g}_t)$ is the sum of the residual interference, media, and electronic noise, whose variance is

$$\sigma_\eta^{2(i)} = \frac{1}{2T} \left(2E_h \sum_{n \neq i} \alpha_n (\mathbf{w}_i^T \mathbf{g}_n)^2 + M_0 \sum_n (\mathbf{w}_i^T \mathbf{g}_n)^2 + N_0 \|\mathbf{w}_i\|^2 \right) / (\mathbf{w}_i^T \mathbf{g}_i)^2. \quad (3.20)$$

Algorithm 2 provides the pseudocode of the proposed soft ITI cancellation algorithm. The inputs to the algorithm are the ADC outputs, the ITI response, the ISI response, and the detection order Π . The output of the algorithm is the set of *a posteriori* LLRs for each track in Π . The algorithm begins by initializing $\alpha_j = 1$ for all tracks j . It then proceeds to the main loop (line 2 – line 11), which steps through each track index in Π . In line 3, the current track index is identified as t , and the set of previously detected tracks that will be used for cancellation is identified as \mathcal{P} in line 4. Observe that line 4 specifically excludes the current track t from \mathcal{P} , which is necessary if track t were detected earlier, since it would not make sense to subtract the contributions from track t when the goal is to detect track t . Observe further from line 4 that the first time through the main loop ($n = 1$), \mathcal{P} will be the empty set. In line 5, the weights are computed using (3.16), and in line 6, the ITI cancellation and suppression is performed using (3.10) and $z_k^{(t)} = y_k^{(t)} / (\mathbf{w}_t^T \mathbf{g}_t)$. After canceling and suppressing the ITI, the result is applied to a 1D detector in line 8; this might be Viterbi, BCJR, or an iterative detector that iterates between a channel detector and an error-control decoder (see Sect. 3.2.3). In line 9, the soft estimates of the ISI symbols are computed. These estimates will be used in the later passes through the main loop to cancel ITI. Finally, in line 10, the reliability measure for these decisions is computed, again for

Algorithm 2: Pseudocode of the Proposed Soft ITI Cancellation Detector

Inputs: ADC outputs $\{\mathbf{r}_k\}$
 Crosstrack and downtrack responses $\{\mathbf{g}_j\}, \{h_k\}$
 Detection order Π
Output: LLR's $\{\lambda_k^{(j)}\}$ for all detected tracks $j \in \Pi$

```

1  Init:  $\alpha_j = 1$  for all  $j$ 
2  for  $n = 1$  to  $|\Pi|$  do
3       $t = \Pi(n)$ 
4       $\mathcal{P} = \{\Pi(1), \dots, \Pi(n-1)\} \setminus \{t\}$ 
5       $\mathbf{w}_t = \left( \mathbf{g}_t \mathbf{g}_t^T + \sum_{j \neq t} (\alpha_j 2E_h + TM_0) \mathbf{g}_j \mathbf{g}_j^T + N_0 \mathbf{I} \right)^{-1} (2E_h + TM_0) \mathbf{g}_t$ 
6       $z_k^{(t)} = \mathbf{w}_t^T \left( \mathbf{r}_k - \sum_{j \in \mathcal{P}} \tilde{x}_k^{(j)} \mathbf{g}_j \right) / (\mathbf{w}_t^T \mathbf{g}_t)$  for all  $k$ 
7      Compute  $\sigma_\eta^2$  using (3.20)
8       $\{\lambda_k^{(t)}\} = \text{Detect}(\{z_k^{(t)}\}, \{h_k\}, \sigma_\eta^2)$ 
9       $\{\tilde{x}_k^{(t)}\} = \text{Convolve}(\{\tanh(\lambda_k^{(t)}/2)\}, \{h_k\})$ 
10      $\alpha_t = E((x_k^{(t)} - \tilde{x}_k^{(t)})^2) / E_x$ 
11 end

```

using in the later passes through the main loop.

3.2.3 Numerical Example

We evaluate the performance of the proposed detector by simulating the model (3.3) in the special case where there are $N = 5$ read heads and $L = 9$ inputs. We assume that the tracks are coded independently by a rate-0.9 regular LDPC code of length 36409 and column weight 3, constructed using the progressive edge growth method [44]. The per-bit SNR for track t , ignoring the ITI from other tracks and after accounting for the rate- R code, can be computed from (3.3) as

$$\text{SNR}_t = \frac{E_h/R}{N_0/\|\mathbf{g}_t\|^2 + M_0}. \quad (3.21)$$

In the downtrack direction we assume the E2PR2 ISI response $\mathbf{h} = [h_0, \dots, h_4] = [1, 4, 6, 4, 1]$. We further assume that the five read heads are centered over five adjacent tracks, and that each reader exhibits the same Gaussian crosstrack response, namely, $\mathbf{g}_5^T = [\mathbf{g}_{1,5}, \dots, \mathbf{g}_{5,5}] = [0.0183, 0.3679, 1, 0.3679, 0.0183]$ and $\mathbf{g}_{5-j}^T = [\mathbf{g}_{1+j,5}, \dots, \mathbf{g}_{5+j,5}]$ for $|j| \leq 4$.

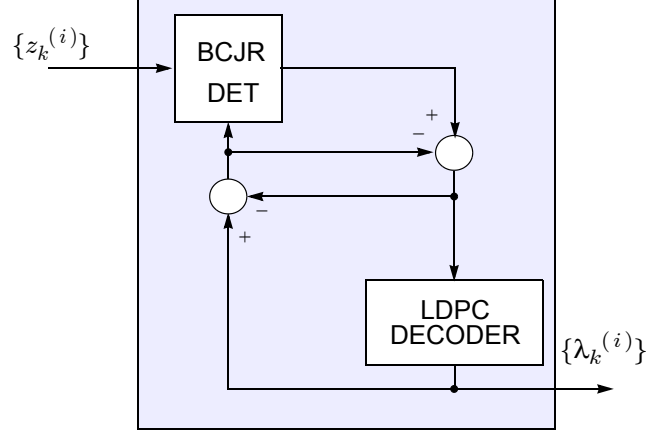


Figure 3.1: Iterative detector based on turbo equalization [36].

The 1D detector of line 8 is implemented using the iterative detector shown in Fig. 3.1, in which a BCJR soft-output channel detector iterates with a soft-output LDPC decoder according to the turbo equalization principle [36]. Note that the BCJR detector does not exploit the data-dependence of the media noise; we expect improved performance using a pattern-dependent noise-predictive BCJR [4]. We implement 10 inner iterations (inside the LDPC decoder) for each outer iteration of the turbo equalizer. We apply the termination criterion from [45]: the iterative process continues as long as $\sum_k |\lambda_k^{(t)}|$ increases, and it stops as soon as it decreases. We consider first the performance in the absence of media noise ($\sigma_j^2 = 0$), so that the relative media noise power $\gamma = M_0/(N_0 + M_0)$ reduces to $\gamma = 0$. The resulting frame error rate (FER) performance is shown in Fig. 3.2 for three different detectors: 1) the linear detector; 2) a hard ITI cancellation detector; and 3) the soft ITI cancellation detector.

First, we focus on the case when all tracks are detected using linear ITI suppression, which are the dashed gray curves in the figure. The two outer tracks¹ (i.e., tracks ± 2) never stray from an FER near unity over the range of SNR values shown; to achieve $\text{FER} < 10^{-2}$ for the outer tracks requires $\text{SNR}_0 = 40$ dB (not shown). The middle track (track 0) achieves $\text{FER} = 10^{-3}$ at $\text{SNR}_0 = 13.6$ dB, while the other two inner tracks (tracks ± 1) require 15.4

¹For convenience we renumber the tracks: track 0 is the center track, tracks ± 1 are its neighbors, etc.

dB, a 1.8 dB difference.

Fig. 3.2 also includes the results of the proposed soft cancellation detector, which are the solid black curves in the figure. These results are based on a detection order of $\mathbf{\Pi} = [0, 1, -1, 0, 1, 2, 1, 0, -1, -2, -1, 0]$, so that the middle track is detected first and last, with all remaining tracks detected at least once along the way (among other candidate detection orders we considered, this performed the best. The problem of optimizing $\mathbf{\Pi}$ to maximize performance is an open problem). The performance for the middle track is only 0.2 dB better with soft ITI cancellation than with linear suppression: the middle track (track 0) with soft ITI cancellation achieves $\text{FER} = 10^{-3}$ at $\text{SNR}_0 = 13.4$ dB. However, soft ITI cancellation improves performance for the remaining tracks. In particular, the two inner tracks (tracks ± 1) perform identically to the middle track (track 0) with soft ITI cancellation. Thus, for the two inner tracks, the soft ITI cancellation detector outperforms the linear detector by 2 dB. The improvement from soft cancellation is even more dramatic for the outer tracks: with soft ITI cancellation, the outer tracks (tracks ± 2) achieve $\text{FER} = 10^{-3}$ at $\text{SNR}_0 = 17.6$ dB, which is only 4.2 dB worse than the center track, and over 22 dB better than can be achieved with linear detection alone. These results suggest that, at least in this one example, the advantage of the soft ITI cancellation strategy is not so much its performance for the inner tracks, but rather its advantage is its ability to reliably recover data from more tracks than is otherwise possible.

The solid gray curves in Fig. 3.2 show the performance of a *hard* ITI cancellation detector, which mimics the proposed soft ITI cancellation detector except with hard decisions $\tilde{a}_k^{(j)} = \text{sign}(\lambda_k^{(j)})$ used in place of soft decisions $\tilde{a}_k^{(j)} = \tanh(\lambda_k^{(j)}/2)$. The hard cancellation detector performs about 0.2 dB worse than the soft cancellation detector for the three inner tracks. Although the benefit of using soft decisions for ITI cancellation is modest in this example, the fact that the benefit comes at essentially no cost in complexity makes it attractive nonetheless.

Next, we consider the performance when the media noise accounts for 80% of the total

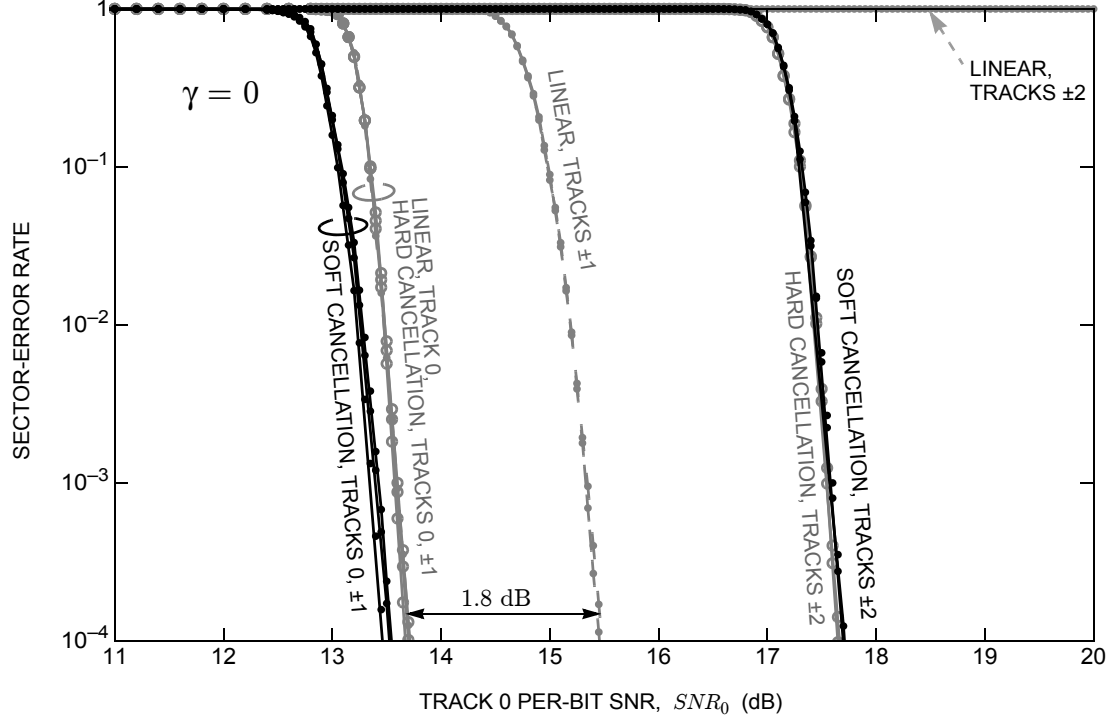


Figure 3.2: FER performance in the absence of media noise ($\gamma = 0$).

noise power, so that $\gamma = M_0/(N_0 + M_0) = 0.8$. All other parameters remain the same as before. The resulting performance is shown in Fig. 3.3.

Consider first the linear detector: Similar to the case with $\gamma = 0$, we observe that the two outer tracks are not recovered reliably for the range of SNR values shown. We also observe that the middle track (track 0) and the other two inner tracks (tracks ± 1) achieve $\text{FER} = 10^{-3}$ at $\text{SNR}_0 = 13$ dB and 15.2 dB, respectively. With soft ITI cancellation, however, we can recover the two outer tracks with $\text{FER} = 10^{-3}$ at $\text{SNR}_0 = 19.4$ dB, the two inner tracks at $\text{SNR}_0 = 14.2$ dB, and the middle track at $\text{SNR}_0 = 12.7$ dB. The performance gain for the soft cancellation detector thus depends on the track: it is a modest 0.3 dB for track 0, but it grows to 1.0 dB for tracks ± 1 .

A comparison of Figs. 3.2 and 3.3 reveals insight into how the two detection strategies react to media noise. In particular, as the media noise ratio increases from $\gamma = 0$ to $\gamma = 0.8$, we observe the following:

1. For the linear detector, the penalty for tracks ± 1 (relative to track 0) jump from 1.8

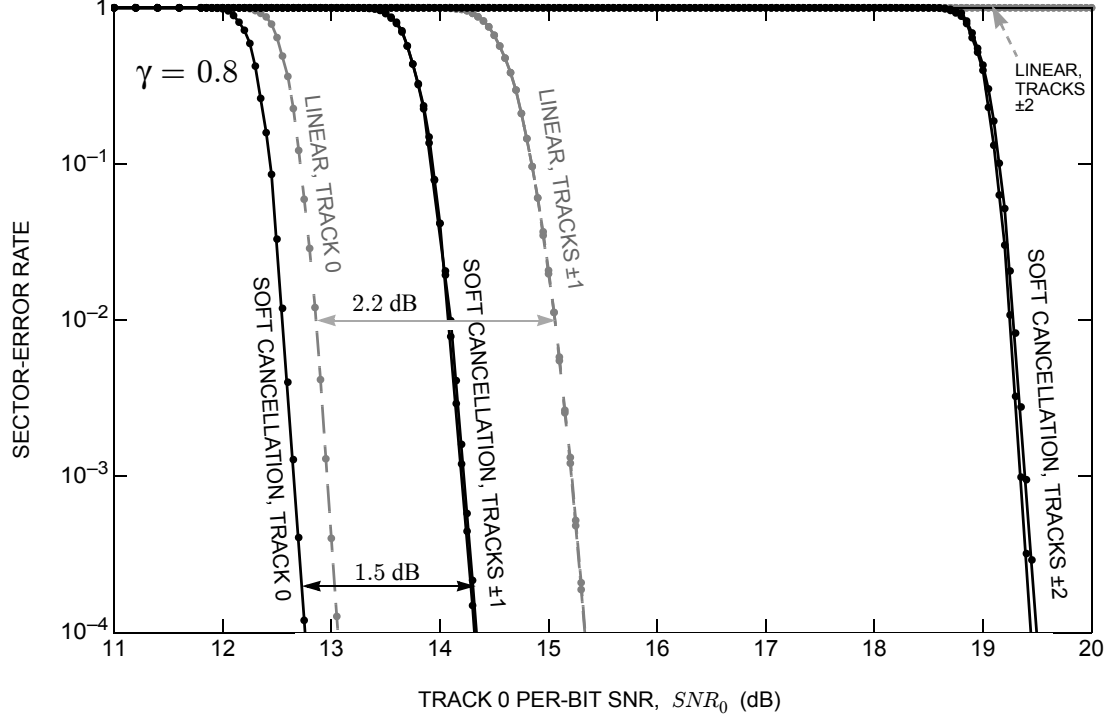


Figure 3.3: FER performance when media noise is dominant ($\gamma = 0.8$).

dB to 2.2 dB.

2. For the soft ITI canceller, the penalty for tracks ± 1 (relative to track 0) jump from 0 dB to 1.5 dB.
3. For the soft ITI canceller, the penalty for tracks ± 2 (relative to track 0) jump from 4.2 dB to 6.7 dB.

The increased penalties in the face of media noise can be explained in part by the absence of any mechanism in the proposed algorithm for estimating and canceling the interference caused by media noise from interfering tracks.

3.3 Summary

We considered the problem of how to process the readback waveforms coming from an array of two or more read heads in a TDMR application. We presented two strategies for mitigating the ITI, one based on linear suppression and one based on soft cancellation.

Numerical results demonstrated that the relative advantage of the two detection strategies depends on the location of the track being detected. For the inner tracks near the center of the read-head array, the gain of the soft ITI cancellation detector over the linear detector is modest. For the outer tracks near the edge of the array, in contrast, the gain of the soft ITI cancellation detector over the linear detector is significant. Therefore, for a given pass of a read-head array, the soft ITI cancellation strategy has demonstrated its ability to reliably recover data from more tracks than would otherwise be possible using linear ITI suppression. Future work should develop adaptive implementations for these algorithms and explore the optimization problem for the detection order **II**.

From the next chapter forward, we add the synchronization challenge to the mix. First in Chapter 4, we use the proposed detection strategies of this chapter plus other established detection strategies for single-track detection of *asynchronous* tracks. We propose to reduce the computational complexity of the established strategies by moving the synchronization task *after* the equalization to a PR channel. Later, in Chapter 5, we will propose ROTAR algorithm for multitrack detection of asynchronous tracks.

CHAPTER 4

SYNCHRONIZATION FOR SINGLE-TRACK DETECTION

The objective here is to tackle the synchronization problem for single-track detection. We assume asynchronous tracks with different phase and frequency offsets, meaning that neither the bit boundaries (phase) nor the bit rates (frequency) are aligned between neighboring tracks. We are interested to study the problem of mitigating ITI in detecting a single track of interest, when the contributing tracks are asynchronous to each other. In particular, we are interested to know which task comes first, the synchronization or ITI mitigation. If synchronization *precedes* the ITI mitigation, it means we need to individually synchronize each readback waveform to the track of interest *before* we can suppress ITI and detect that particular track. In contrast, if synchronization *follows* the ITI mitigation, it means we can synchronize only once to the track of interest *after* suppressing ITI from other tracks. The answer to this question strongly impacts the computational cost of the overall detection process: If we can efficiently synchronize for a track of interest *after* suppressing the ITI, we will only need one synchronization loop for every track of interest instead of one synchronization loop for every readback waveform or every ADC.

In a first attempt towards synchronization for TDMR, in Sect. 4.1, we study the synchronization for the separable channel model of (3.3) with added timing offsets. Later, in Sect. 4.2, we replace the separable channel with a realistic non-separable channel using the readback waveforms provided by Ehime University [46]. We first explain synchronization over the separable channel.

4.1 Synchronization Over Separable Channel

Using the two detection strategies of Chapter 3, we aim at detecting asynchronous tracks. We consider asynchronous tracks having different frequency and phase offsets. Therefore, if

$1/T$ denotes the ADC sampling rate, every track j with frequency offset parameter ΔT_j has a bit rate of $1/(T + \Delta T_j)$, and a phase offset of τ_j . The underlying continuous readback waveform of (3.1), therefore, becomes

$$r_i(t) = \sum_j g_{i,j} \left(\sum_\ell a_\ell^{(j)} h(t - \ell(T + \Delta T_j) - \tau_j) + (a_\ell^{(j)} - a_{\ell-1}^{(j)}) j_\ell^{(j)} q(t - \ell(T + \Delta T_j) - \tau_j) \right) + n_i(t), \quad (4.1)$$

where $h(t)$ is the bit response whose bandwidth is half the bit rate.¹ As before, we apply an antialiasing filter to (4.1) and sample at the ADC rate yielding:

$$r_k^{(i)} = \sum_j g_{i,j} \left(\sum_\ell a_\ell^{(j)} \sum_m h_m f((k - \ell - m)T - \ell\Delta T_j - m\Delta T_j - \tau_j) + (a_\ell^{(j)} - a_{\ell-1}^{(j)}) j_\ell^{(n)} q(t - \ell(T + \Delta T_j) - \tau_j) \right) + n_k^{(i)}, \quad (4.2)$$

where $f(t) = \frac{\sin(\pi t/(T+\Delta T_j))}{\pi t/(T+\Delta T_j)}$. For sufficiently small ΔT_j , this simplifies to:

$$r_k^{(i)} \approx \sum_j g_{i,j} \left(\sum_m h_m \alpha_{k-m}^{(j)} + \sum_m q_m \beta_{k-m}^{(j)} \right) + n_k^{(i)}, \quad (4.3)$$

where

$$\alpha_k^{(j)} = \sum_\ell a_\ell^{(j)} f(kT - \ell(T + \Delta T_j) - \tau_j) \quad (4.4)$$

and

$$\beta_k^{(j)} = \sum_\ell (a_\ell^{(j)} - a_{\ell-1}^{(j)}) j_\ell^{(j)} f(kT - \ell(T + \Delta T_j) - \tau_j) \quad (4.5)$$

respectively are the delayed ISI symbol and the delayed media noise symbol of track j at time k .

Therefore, if we replace the ISI and the media noise symbols in (3.3), respectively, with

¹Since different tracks, in general, have different bit rates, therefore the bit responses of different tracks have different bandwidths. To avoid notation complexity, however, we avoid adding the superscript (j) for the bit response of every track j .

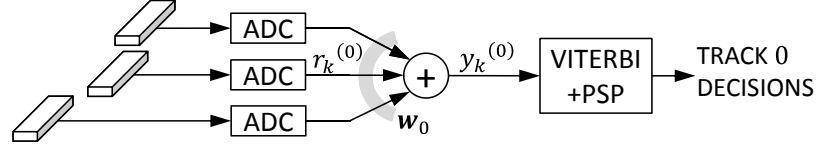


Figure 4.1: Asynchronous linear ITI suppression

(4.4) and (4.5), we have incorporated timing offset into our MIMO description of TDMR channel according to:

$$\mathbf{r}_k = \sum_j \left(\sum_m h_m \alpha_{k-m}^{(j)} + \sum_m q_m \beta_{k-m}^{(j)} \right) \mathbf{g}_j + \mathbf{n}_k, \quad (4.6)$$

In order to detect every track from the set of waveforms above, we applied our two detectors of Chapter 3: 1) linear combining, and 2) soft ITI cancellation detectors.

4.1.1 Asynchronous Linear ITI Suppression

First, we apply the linear detector. The linear combining weights of (3.6) in Sect. 3.2.1, where there is no timing offset, depends on the bit period T that is the same as the ADC sampling period. With frequency offset, here, however, the bit period for every track j is $T_j = T + \Delta T_j$ that is no longer the same as the ADC sampling period T . Therefore, the linear combining MMSE weight for detecting track t becomes

$$\mathbf{w}_t = \left(\sum_j \mathbf{g}_j \mathbf{g}_j^T + \left(\frac{N_0}{2E_h + T_j M_0} \right) \mathbf{I} \right)^{-1} \mathbf{g}_t. \quad (4.7)$$

Nevertheless, in practice, the frequency offset parameters $\{\Delta T_j\}$ are quite small. Therefore, at least theoretically, it seems reasonable to ignore the frequency offset and apply the same exact weights of (3.6) in order to suppress the ITI in the new asynchronous waveforms of (4.6). Therein, we choose to *asynchronously* suppress ITI from the adjacent tracks before we synchronize and detect every track. An example of asynchronous linear ITI suppression with $N = 3$ read heads is shown in Fig. 4.1. Also, as Fig.4.1 shows, the synchronization

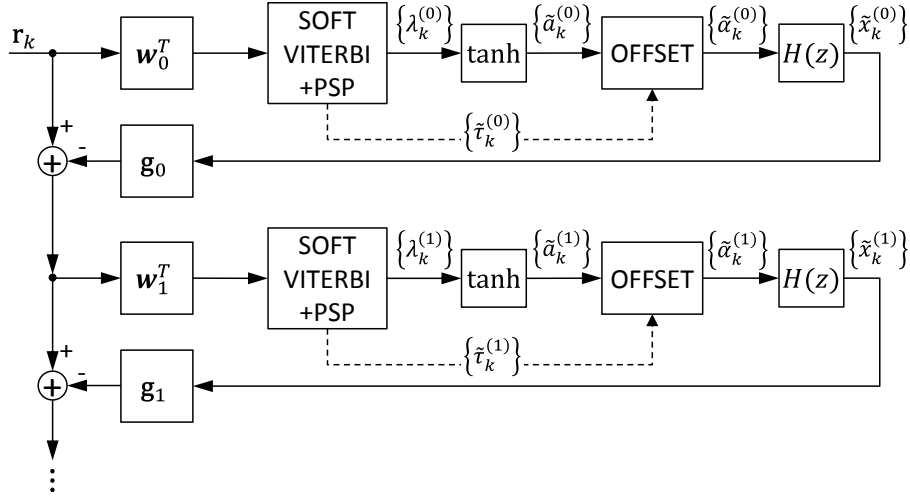


Figure 4.2: Synchronous ITI cancellation

task is embedded inside a 1D Viterbi detector using a PSP for timing recovery.

4.1.2 Synchronous ITI Cancellation

Second, we apply the soft ITI cancellation detector according to Fig. 4.2. Similar to the linear combining, the ITI suppression weights for detecting track t considering the frequency offset parameters $\{\Delta T_j\}$ becomes

$$\mathbf{w}_t = \left(\sum_j \left(\frac{\alpha_j 2E_h + T_j M_0}{2E_h + T_j M_0} \right) \mathbf{g}_j \mathbf{g}_j^T + \left(\frac{N_0}{2E_h + T_j M_0} \right) \mathbf{I} \right)^{-1} \mathbf{g}_t \quad (4.8)$$

Also similar to the linear combining detector and considering that $\{\Delta T_j\}$ are small, we choose to ignore the timing offsets in calculating the weights. Fig. 4.2 shows how ITI cancellation detector is used to detect asynchronous tracks from the waveforms of (4.6). To detect every track t , here, we first *asynchronously* suppress ITI by applying the MMSE weights of (4.8). Next, we detect every track t using a soft-output Viterbi detector with embedded PSP for timing recovery. In order to compute the interference caused by track t , we apply the estimated timing offsets provided by the PSP to the sequence of the soft estimates of the bits on track t and modulated the result with the ISI response. This way we can *synchronously* cancel the interference caused by each track on every other track which is going to be detected afterwards.

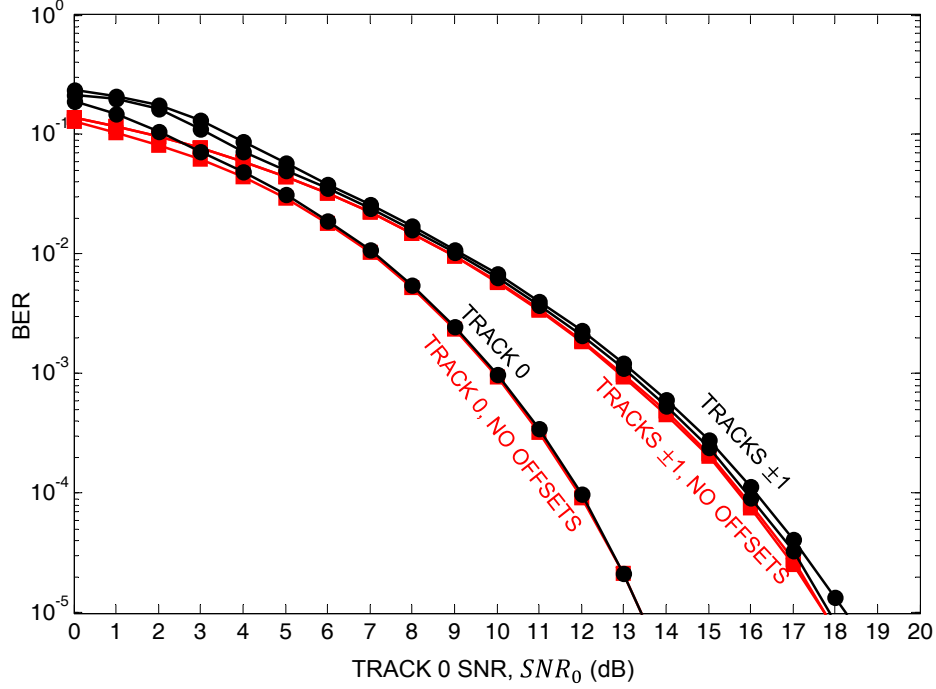


Figure 4.3: BER performance of synchronous ITI cancellation detector (the same as that of asynchronous ITI suppression) in the presence and absence of timing offsets (for convenience we renumber the tracks: track 0 is the middle track and tracks ± 1 are its neighbors).

4.1.3 Numerical Example

To verify the proposed detectors of asynchronous ITI suppression (Fig. 4.1) and synchronous ITI cancellation (Fig. 4.2), we simulated for a case of $N = 3$ read heads each centered on their own track, a downtrack response of $\mathbf{h} = [1, 0.6, 0.2]$ and a crosstrack response of $\{g_{i,j}\}_{1 \leq i \leq K=7, 1 \leq j \leq N=3} \in \begin{bmatrix} 0.0025 & 0.223 & 1 & 0.223 & 0.0025 & 0 & 0 \\ 0 & 0.0025 & 0.223 & 1 & 0.223 & 0.0025 & 0 \\ 0 & 0 & 0.0025 & 0.223 & 1 & 0.223 & 0.0025 \end{bmatrix}$. The frequency offset parameters and the phase offsets of the $K = 7$ contributing tracks respectively were $[\Delta T_1, \Delta T_2, \Delta T_3, \Delta T_4, \Delta T_5, \Delta T_6, \Delta T_7]/T = [200, 300, 50, 100, 160, 200, 300]ppm$ and $[\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6, \tau_7]/T = [0.11, 0.81, 0.73, 0.79, 0.32, 0.51, 0.29]$. Also, all simulations were performed in a dominant media noise environment with $\gamma = 0.8$. The set of black curves in Fig. 4.3 show the BER performance of only the synchronous ITI cancellation detector, since in this example, the two detectors performed almost the same. Fig. 4.3 also shows the performance of the two detectors in the absence of timing offsets, shown as red curves marked with “NO OFFSETS”. In the absence of timing offsets, the two detectors boil down

to the linear combining and soft ITI cancellation detectors of Chapter 3. We observe that there is a close match between the cases of detecting asynchronous tracks and synchronous tracks (with no offsets).

Therefore, we conclude that the proposed detectors were effective in suppressing ITI asynchronously before synchronizing and detecting every track of interest, without any loss in performance. This is important because we were able to suppress ITI before applying one synchronizing loop for every track of interest and therein largely save in computational complexity.

4.2 Synchronization Over Nonseparable Channel

So far, we have considered a separable channel. A realistic model for TDMR, however, is highly nonseparable. The nonseparable MIMO model which includes frequency and phase offsets is similarly derived from (4.1)-(4.6) when both crosstrack and downtrack responses are absorbed into a single impulse response $h^{(i,j)}(t)$ from track j to reader i , as follows.

We rewrite equation (4.1) for the nonseparable channel:

$$r^{(i)}(t) = \sum_j \sum_{\ell} a_{\ell}^{(n)} h^{(i,j)}(t - \ell(T + \Delta T_j) - \tau_j) + n^{(i)}(t). \quad (4.9)$$

We filter the i -th readback waveform by a low-pass antialiasing filter and then sample at the ADC rate $1/T$. Further, we apply the practical assumption of sufficiently small ΔT_j to arrive at the MIMO model for the nonseparable channel with frequency and phase offsets:

$$r_k^{(i)} = \sum_j \sum_m h_m^{(i,j)} \alpha_{k-m}^{(j)} + n_k^{(i)}, \quad (4.10)$$

where α 's are defined according to (4.4).

From this section forward, we omit the media noise and mainly focus on the synchronization problem.

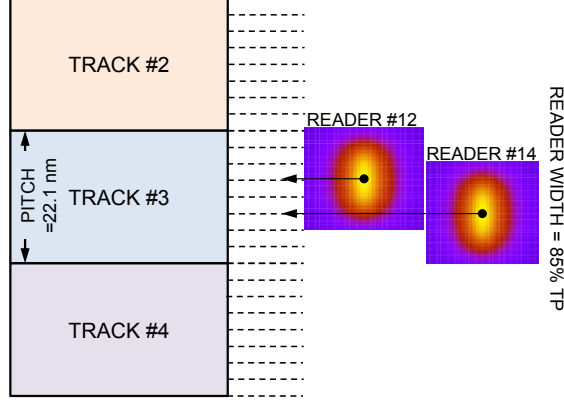


Figure 4.4: Readers positions relative to the data tracks are marked with dashed lines. The two readers (indicated with their responses) were used to extract the channel impulse response. The tracks width (pitch) is 22.1nm , and the reader width is 85% of the track pitch.

We were provided with a set of readback waveforms by Ehime University. The waveforms were generated from a Voronoi model for TDMR channel [46]. These waveforms, however, did not include any timing offsets. Therefore, in order to include timing offsets, we extracted the waveforms' underlying channel responses $\{h^{(i,j)}\}$ and generated new waveforms according to (4.10). The readers positions relative to the data tracks and also the two readers used to extract the channel impulse responses are shown in Fig. 4.4. These readers are selected because they mainly cover the track of interest (the middle track) and they are far from the outer tracks to avoid interference.

The entire read channel including the proposed architecture for detecting the middle track (track 3) from $N = 2$ readback waveforms is presented in Fig. 4.5. Fig. 4.5 (a) is the realization of the MIMO model of (4.10) where α 's are generated according to (4.4) with frequency offset parameters $\{\Delta T_j\}$ and phase offsets $\tau_j = 0, \forall j \in [2, 3, 4]$. Here, the shifted user bits (α 's) are channeled through the estimated vectorized responses $\hat{\mathbf{h}}^{(i,j)} = [\hat{h}_0^{(i,j)}, \dots, \hat{h}_\mu^{(i,j)}]$ with memory μ , and $\forall j \in [2, 3, 4]$ and $\forall i \in [1, 2]$, which include all the significant taps of $\hat{h}^{(i,j)}(t)$ sampled at the sampling rate $1/T$.

Fig. 4.5 (b) is a MISO equalizer ($N = 2$ to 1) that shortens the channel in order to detect the middle track. Here, we use the generalized partial response (GPR) equalization strategy to derive the two equalizer filters and the target [47], [48]. Following a GPR

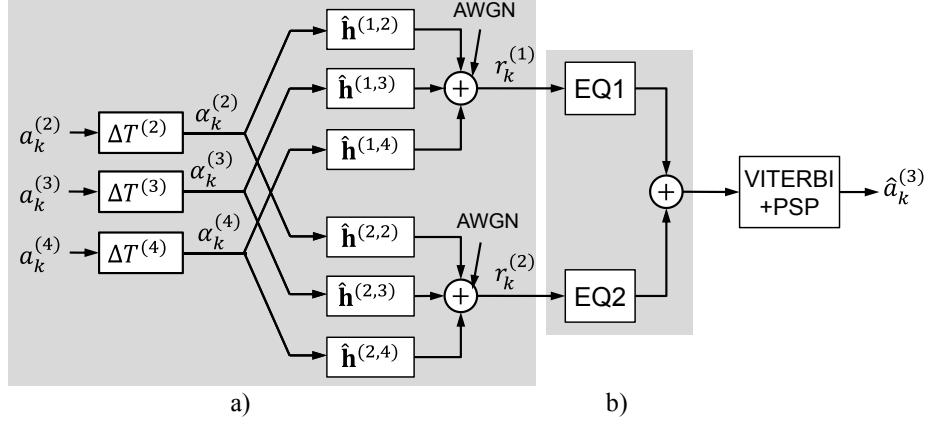


Figure 4.5: The proposed architecture for detecting the middle track: (a) the MIMO model with frequency offset of a nonseparable channel with $N=2$ readers and $K = 3$ tracks, (b) a MISO equalizer to detect the middle track, and (c) a PSP algorithm embedded inside a Viterbi detector

strategy presented in Appendix C, we derive the MMSE solution for joint optimization of N equalizer filters and a target for single-track detection, in the absence of timing offsets. In the presence of timing offsets, however, of course the solution changes. Nevertheless, our key finding is that, for practical values of frequency offset parameters $\{\Delta T_j\}/T$ that are sufficiently small, and for any phase offsets, the change in the MMSE solution is negligible. In fact, the change is so negligible that we can safely conclude that *for practical timing offsets in magnetic recording channel, the MMSE solution for joint optimization of the equalizer filters and the target does not depend on the timing offsets*. In other words, *the resulted equalizer and the target is transparent to timing offsets, as long as the frequency offset parameter for the track of interest is sufficiently small*. Therefore, the timing offsets can be completely ignored in computing the MMSE solution with no performance loss. This result is directly generalizable to MIMO equalization as well, which it will be discussed in the next chapter.

Hence, the MISO equalizer in Fig. 4.5 (b) is the same equalizer that would have been used if all tracks were written synchronously. The MISO equalizer breaks the problem down to a conventional 1D synchronization and detection which can be jointly handled by a PSP algorithm embedded inside a Viterbi detector, as in Fig. 4.5 (c).

4.2.1 Numerical Results

Simulations of the proposed architecture of Fig. 4.5 were performed on $K = 3$ data tracks of length $L = 40950$ bits. The Ehime waveforms were sampled twice the bit rate, therefore, we used the first 2000 samples, corresponding to the first 1000 bits, of the waveforms to compute the equalizer and the target. Since the sampling rate was twice the bit rate, the equalizer filter coefficients are fractionally-spaced in time, and therefore the equalizer is called a fractionally-spaced equalizer (FSE). We jointly optimized the FSE equalizer and the target for every point of the SNR axis according to Appendix C. We then used the entire length of the sector (81900 samples, or 40950 bits) to obtain the BER performance of Fig. 4.6. Frequency offsets were injected into the test waveforms according to Fig. 4.5. The frequency offsets parameters were $[\Delta T_2, \Delta T_3, \Delta T_4]/T = [4000, 200, 20, 100]ppm$.

Fig. 4.6 also shows the performance of the proposed architecture in the absence of timing offsets where $\Delta T_j = 0$. In the absence of timing offsets, the Viterbi plus PSP block in Fig. 4.5 reduces to a standard Viterbi detector. We observe that the performance of the middle track in the presence of timing offsets closely matches the performance in the absence of timing offsets, which clearly certifies that the proposed architecture is capable of detecting asynchronous tracks. We also simulated for different reader geometries and different number of readers and relevant tracks to detect a single track of interest. The results were unanimous in suggesting that architectures similar to Fig. 4.5 where MISO equalization precedes the synchronization and detection can successfully detect the track of interest.

Finally, based on the results provided on nonseparable and separable channel models, we were able to equalize ignoring the presence of timing offsets before synchronizing and detecting the track of interest. As mentioned before, this result implies that we do not need one synchronization loop for every reader. Rather, one synchronization loop is sufficient for every track of interest.

Therefore, the contribution of this thesis for the case of detecting one track at a time, is a notable reduction in complexity which results when synchronization moves *after* the

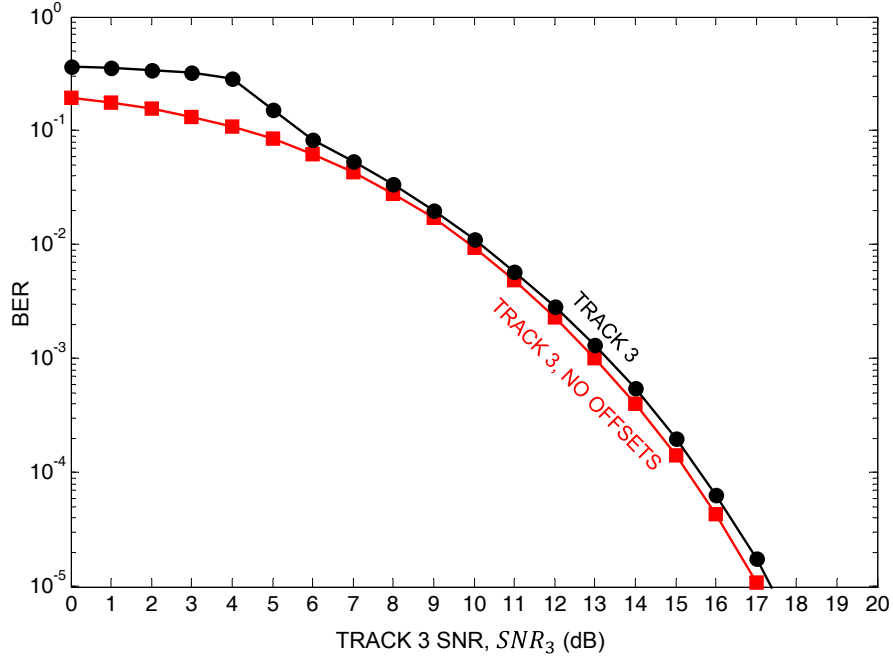


Figure 4.6: BER performance of the proposed architecture of Fig. 4.5 for detecting the center track.

equalization. This is in contrast to the conventional read channels where synchronization *precedes* equalization.

4.3 Summary

In this chapter, we first considered the single-track detection from multiple readback waveforms when the tracks have different frequency and phase offsets. We applied the proposed soft ITI cancellation detector of Chapter 3 to *asynchronously* mitigate ITI before synchronizing for and detecting a single track of interest. Since the proposed detector breaks the problem down to several 1D detection problems, a soft-output Viterbi detector plus PSP for timing recovery was used for synchronization and detection of every track of interest. Numerical results showed that the proposed architecture is capable in mitigating ITI for the separable MIMO model considered.

Next, we considered a more realistic nonseparable MIMO model for TDMR channel. We found that, within working precision, the solution for joint optimization of the equalizer

filters and the target is independent of timing offsets. This finding has important implication: As opposed to the conventional TDMR read channels where equalization is done *after* synchronization and therefore there is one synchronization block required for every reader used, we can effectively equalize *before* synchronizing for and detecting every track of interest, and therefore we only need one synchronization block for detecting every track of interest, regardless of the number of readers used.

CHAPTER 5

SYNCHRONIZATION FOR MULTITRACK DETECTION

Current implementations of TDMR technology use multiple readers for single-track detection. In the previous chapter, we addressed the synchronization problem for single-track detection: The synchronization problem in the single-track setting is straightforward, since off-the-shelf one-dimensional strategies based on a PLL, ITR, or PSP can be applied after the MISO equalizer front end. In this case the equalizer outputs are synchronized to the track of interest, regardless of the timing offsets of the interfering tracks [49]. The result is an instance of modular design, in which the functions of synchronization and detection are implemented separately.

To achieve the full potential of TDMR system, however, the future implementations will embrace ITI by jointly detecting *multiple* adjacent tracks using a *joint* or *multitrack* detector [5]. Within multitrack detection, the synchronization problem drastically changes: *We cannot simultaneously synchronize a readback waveform to multiple tracks which have different timings.* Fig. 5.1 illustrates an example of two overlapping readers scanning two adjacent tracks which have different bit rates (frequency) and bit boundaries (phase). We can clearly see that each reader can only be synchronized to either one of the tracks and not to the both tracks at the same time: being synchronous to one necessarily implies being asynchronous to the other. The implication is that even if the timings of the two tracks are perfectly known, unlike the previous chapter, synchronization and detection can no longer be performed separately, but instead must be performed jointly. There has been no published solution for this problem in the literature so far.

In this chapter, we present the rotating target (ROTAR) algorithm for the joint synchronization and multitrack detection of asynchronous tracks from multiple readback waveforms [50]. First in this chapter, we explain ROTAR considering a *perfectly equalized* channel.

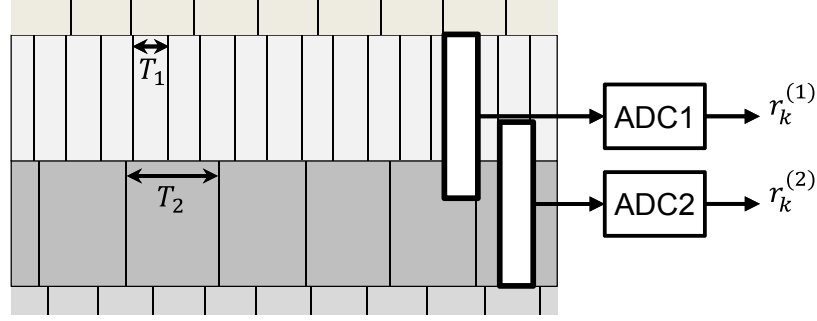


Figure 5.1: An example of two tracks of interest whose timing differ in frequency and phase, and two readers with significant overlap.

Next in Chapter 6, we will present our equalization strategy to precede the ROTAR algorithm.

5.1 Channel Model and Assumptions

We consider the problem of jointly detecting K tracks from N readback waveforms. We assume a perfectly equalized partial response channel with independent timing offsets for each of the K tracks, so that the readback waveform from the i -th of N read heads is:

$$r_i(t) = \sum_{j=1}^K \sum_n a_n^{(j)} h_{i,j}(t - nT - \tau_n^{(j)}) + n_i(t) \quad (5.1)$$

where $a_n^{(j)} \in \{\pm 1\}$ is the n -th bit of track $j \in \{1, \dots, K\}$, $h_{i,j}(t)$ is the bit response at reader i from track j , assumed to be bandlimited to half the bit rate, $\tau_n^{(j)} \geq 0$ is the timing offset for the n -th bit of track j , defined relative to the ADC sampling period T , and $n_i(t)$ is the additive noise for the i -th read head. We assume independent white and Gaussian noise with power-spectral density $N_0/2$ for each of the read heads. The assumption that the $\{\tau_n^{(j)}\}$ be nonnegative is equivalent to an assumption that the ADC sampling rate is large enough to avoid signal aliasing.

The i -th readback waveform is filtered by a low-pass antialiasing filter and then sampled

at the ADC rate $1/T$, yielding

$$r_k^{(i)} = \sum_{j=1}^K \sum_n a_n^{(j)} h_{i,j}(kT - nT - \tau_n^{(j)}) + n_k^{(i)}, \quad (5.2)$$

where $n_k^{(i)}$ is the k -th sample of the filtered noise $n_i(t)$, with zero mean and variance $N_0/(2T)$. Collecting the N samples from each of the N read heads at time k into the vector $\mathbf{r}_k = [r_k^{(1)}, \dots, r_k^{(N)}]^T$, and using (5.2), we arrive at a MIMO model:

$$\mathbf{r}_k = \sum_{j=1}^K \sum_n a_n^{(j)} \mathbf{h}_j(kT - nT - \tau_n^{(j)}) + \mathbf{n}_k, \quad (5.3)$$

where $\mathbf{h}_j(t) = [h_{1,j}(t), h_{2,j}(t), \dots, h_{N,j}(t)]^T$ is the vector-valued bit response (across all N readers) for track j , and $\mathbf{n}_k = [n_k^{(1)}, n_k^{(2)}, \dots, n_k^{(N)}]^T$.

5.2 Detection Algorithms

5.2.1 The Case of A Single Isolated Track

Before attacking the general problem of detecting multiple asynchronous tracks from multiple readback waveforms, we first examine the simpler case of detecting a single isolated track ($K = 1$) from a single readback waveform ($N = 1$) of the form $r(t) = \sum_n a_n h(t - nT - \tau_n) + n(t)$, where a_n is the n -th bit of the track, $h(t)$ is the bit response whose bandwidth is equal to half the bit rate, and τ_n is the timing offset of the n -th bit. To be concrete, we will assume a constant frequency offset here, so that $\tau_n = n\Delta T$, where ΔT is the frequency offset parameter. Sampling at the ADC rate $1/T$ yields:

$$r_k = r(kT) = \sum_n a_n h(kT - nT - \tau_n) + n_k. \quad (5.4)$$

In the following we describe two strategies for implementing the maximum-likelihood (ML) sequence detector: 1) the conventional modular strategy, and 2) an alternative strategy.

The latter strategy will eventually be generalized and adopted for the multiple-track scenario.

1) The Conventional Modular Strategy

The usual modular approach is to separately synchronize and detect. This is illustrated in the lower branch of Fig. 5.2. First, the ADC samples $\{r_k\}$ are passed to an ITR block, which aims to recover the readback samples that would have arisen were the readback waveform sampled at the correct sampling times, resulting in:

$$\begin{aligned}\hat{r}_k &= r(kT + \tau_k) \\ &= \sum_n a_n h(kT - nT - \tau_n + \tau_k) + \hat{n}_k \\ &\approx \sum_n a_n h(kT - nT - \tau_k + \tau_k) + \hat{n}_k \\ &= \sum_{\ell=0}^{\mu} h_{\ell} a_{k-\ell} + \hat{n}_k,\end{aligned}\tag{5.5}$$

where the approximation (with τ_n replaced by τ_k) is valid when the timing offset varies slowly enough that it is approximately constant over the duration for which the target $h(t)$ is significant, where $\{h_k = h(k(T + \Delta T))\}$ is the bit response sampled at the bit rate, and where the interpolated noise $\{\hat{n}_k\}$ has the same statistics as the original $\{n_k\}$. For convenience we assume a causal target $\mathbf{h} = [h_0, h_1, \dots, h_{\mu}]^T$ with memory μ , so that $h_k = 0$ for both $k < 0$ and $k > \mu$. After synchronization, the interpolated samples of (5.6) may then be passed to a 2^{μ} -state Viterbi detector, designed for the target \mathbf{h} .

2) An Alternative Strategy

Rather than resampling the ADC outputs via interpolation, however, an alternative approach would be to feed them directly to a detector that internally accounts for the asynchrony, as illustrated in the upper branch of Fig. 5.2. (This is the approach taken by the ROTAR algorithm introduced in Sect. 5.2.2.) This is possible because we can approximate

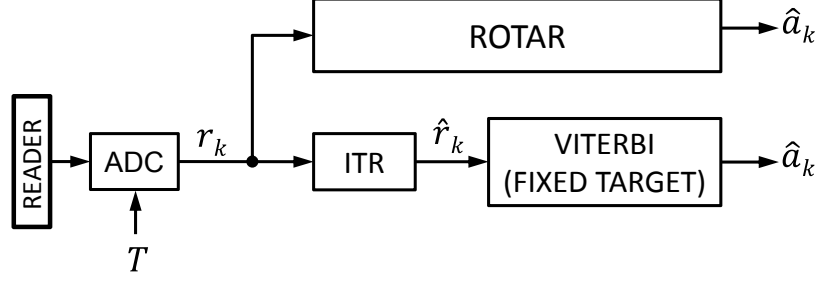


Figure 5.2: Conventional modular (lower branch) versus alternative (upper branch) strategy in synchronization and detection of a single isolated track.

the noiseless ADC output $s_k = r_k - n_k$ from (5.4) as the convolution of the bit sequence $\{a_k\}$ with a *time-varying* impulse response, as derived below:

$$\begin{aligned}
 s_k &= \sum_n a_n h(kT - nT - \tau_n) \\
 &\approx \sum_n a_n h(kT - nT - \tau_k)
 \end{aligned} \tag{5.7}$$

$$\approx \sum_{\ell=-M/2}^{\mu+M/2} h(\ell T - \tau_k) a_{k-\ell}, \tag{5.8}$$

where the approximation in the second line is the same as in (5.5). As a sanity check, the time-varying convolution in (5.8) reduces to the time-invariant convolution in (5.6) for the special case when $\tau_k = 0$ for all k , i.e., for the special case when the ADC is synchronized to the bit rate. In that case, the limits of the last sum in (5.8) range from $\ell = 0$ to μ . In contrast, in the general case when the ADC is not synchronized, the limits of the sum would in principle extend from $\ell = -\infty$ to ∞ for a bandlimited bit response. In practice, however, there will only be a small number of terms that contribute significantly to the sum. To account for this, we introduce a new variable M , which we assume to be even, representing the extra memory used to represent the time-varying impulse response, beyond the memory μ of the original target. The second approximation in (5.8) is because M is finite, and is accurate for even moderate choices of M .

An example of a time-varying target is shown in Fig. 5.3, assuming a frequency offset of

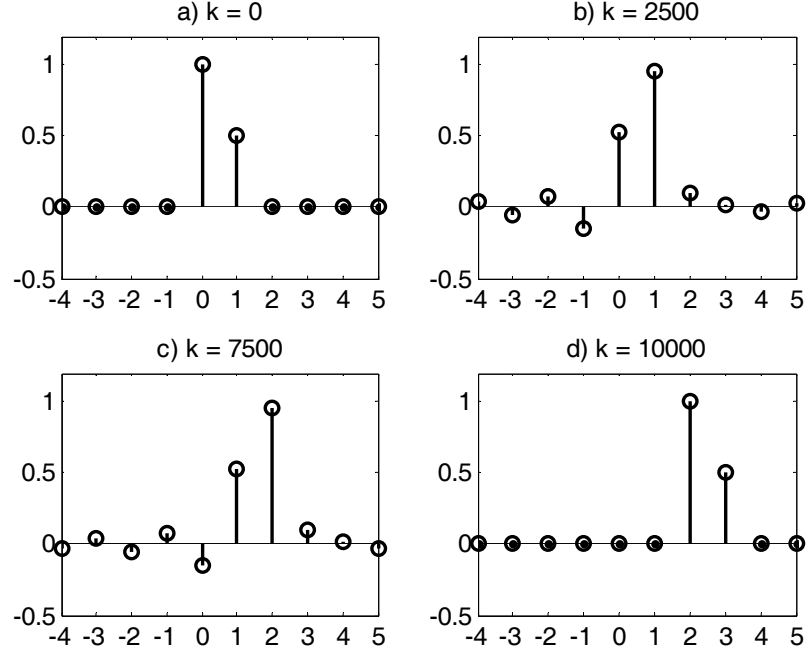


Figure 5.3: An illustration of a moving target for the case of frequency offset with $\Delta T/T = 2 \times 10^{-4}$, and a sector length $L = 10^4$, assuming $M = 8$: (a) The target $\mathbf{h}[0]$ at time $k = 0$; (b) the target $\mathbf{h}[2500]$ at one quarter of the sector; (c) the target $\mathbf{h}[7500]$ at three quarters of the sector; and (d) the target $\mathbf{h}[10000]$ at the end of the sector.

$\tau_k = k\Delta T$ with $\Delta T/T = 2 \times 10^{-4}$, and a sector of length $L = 10^4$ bits. The extra memory in this illustration is $M = 8$. At the beginning of the sector (Fig. 5.3a), the resampled target is a zero-padded version of the synchronous target $\mathbf{h} = [h_0, h_1] = [1, 0.5]$, with only two nonzero taps. As we move forward through the sector, the target drifts to the right and the number of nonzero taps increases. At one quarter of the way through the sector (Fig. 5.3b), the target is shifted by $\tau_k/T = 0.5$ bit periods to the right, and clearly has more than two significant taps. Likewise at three quarters of the way through the sector (Fig. 5.3c), where the target has shifted by $\tau_k/T = 1.5$ bit periods, there are more than two significant taps. By the end of the sector (Fig. 5.3d), the target has shifted by two full bit periods, and again has only two nonzero taps. With the aid of (5.8), the unsynchronized ADC output may be viewed as the output of a time-varying finite-state machine with independent noise, so that the ML detector can be implemented by a $2^{\mu+M}$ -state Viterbi algorithm based on the

time-varying target

$$\mathbf{h}[k] = [h(-MT/2 - \tau_k), \dots, h((\mu + M/2)T - \tau_k)]^T$$

with memory $\mu + M$. The time-varying target prevents us from precomputing the expected outputs for each state transition in the trellis; instead they must be computed anew at each stage according to the convolution in (5.8).

The example of Fig. 5.3 seems to suggest that the amount of extra memory M required to accommodate the moving target will depend on not only the severity of the frequency offset but also the length of the sector. The extra memory is a significant drawback because it increases the number of states and therefore the complexity of the detector. Fortunately there is an efficient strategy for significantly reducing the memory requirements, regardless of the frequency offset parameter and the sector length, as described in Sect. 5.2.2.

3) Numerical Results

We examine the alternative strategy by using the time-varying convolution of (5.8) to model the readback waveform, according to:

$$r_k = \sum_{\ell=-M/2}^{\mu+M/2} h(\ell T - \tau_k) a_{k-\ell} + n_k, \quad (5.9)$$

where the time-varying target is the same used in Fig. 5.3, $\mathbf{h} = [1, 0.5]$, assuming a frequency offset of $\tau_k = k\Delta T$ with $\Delta T/T = 2 \times 10^{-4}$, and a sector of length $L = 10^4$ bits. We implement the upper branch of Fig. 5.2 using a Viterbi algorithm where the expected output for every transition (p, q) is computed anew for time k , according to:

$$o_k(p, q) = \sum_{\ell=-M/2}^{\mu+M/2} a_{k-\ell}(p, q) h(\ell T - \tau_k),$$

where $\{a_k(p, q)\}$ are the bits on the survivor path which arrives at the transition from state

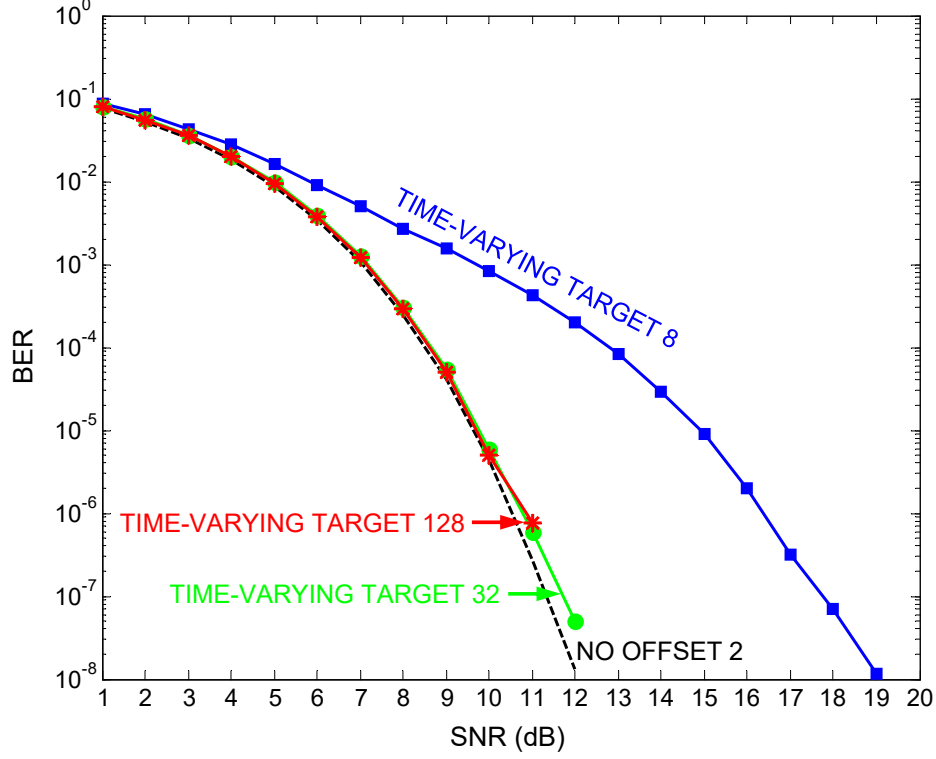


Figure 5.4: BER performance of the alternative strategy with time-varying target based on $h = [1, 0.5]$, in detecting a single isolated track from a single readback waveform of (5.9).

p at time k to state q at time $k + 1$. The branch metric for this transition (p, q) at time k is $\gamma_k(p, q) = |r_k - o_k(p, q)|^2$.

Fig. 5.4 shows the BER performance results of the alternative strategy with different values of extra memory parameter M . The curve labeled “TIME-VARYING TARGET 8” shows the performance of a $2^{\mu+M} = 2^{1+2}$ -state Viterbi algorithm whose memory parameter is $M = 2$. Similarly, the curves labeled “TIME-VARYING TARGET 32” and “TIME-VARYING TARGET 128” show the performance of $2^{\mu+M} = 2^{1+4}$ -state and $2^{\mu+M} = 2^{1+6}$ -state Viterbi algorithm with $M = 4$ and $M = 6$, respectively.

Also shown in Fig. 5.4 is the performance of a fictitious system for which the track was written synchronously with the ADC sampling rate. Therefore a standard Viterbi algorithm with $2^\mu = 2^1$ states can detect this synchronous case. The performance is represented by the dashed line labeled with “NO OFFSET 2”.

We observe that the alternative strategy based on a time-varying target with $M = 4$ and

also $M = 6$ closely match the performance of the synchronous system with no timing offset. This means that a choice of $M = 4$ can sufficiently capture the movement of the target throughout the sector. Nevertheless, the high complexity due to the extra memory added is the drawback of the alternative strategy that will be addressed in Sect. 5.2.2.

5.2.2 Joint Detection of Multiple Asynchronous Tracks

Here, we turn our focus back to the joint detection of K tracks from N readback waveforms. Unlike the case of the isolated track of the previous section, we can no longer separately synchronize and detect in a modular way. Nevertheless, the alternative strategy based on a time-varying target perfectly suits our purpose. Applying the time-varying convolution approximation from (5.8) separately to each track's contribution to reader i , the vector of readback samples from (5.3) can be written as:

$$\mathbf{r}_k \approx \sum_{j=1}^K \sum_{\ell=-M_j/2}^{\mu+M_j/2} \mathbf{h}_j(\ell T - \tau_k^{(j)}) a_{k-\ell}^{(j)} + \mathbf{n}_k, \quad (5.10)$$

where M_j is the extra memory parameter assigned to track j . This is a noisy output of a finite-state machine, so that the K tracks can be jointly detected using the Viterbi algorithm. The expected outputs for a transition (p, q) at time k are:

$$\mathbf{o}_k(p, q) = \sum_{j=1}^K \sum_{\ell=-M_j/2}^{\mu+M_j/2} a_{k-\ell}^{(j)}(p, q) \mathbf{h}_j(\ell T - \tau_k^{(j)}), \quad (5.11)$$

where $\{a_k^{(j)}(p, q)\}$ are the bits of track j on the survivor path which arrives at the transition from state p at time k to state q at time $k + 1$. The branch metric for this transition (p, q) at time k is $\gamma_k(p, q) = \|\mathbf{r}_k - \mathbf{o}_k(p, q)\|^2$.

1) High Complexity

The complexity of the joint Viterbi algorithm is dominated by the number of states

$\prod_{j=1}^K 2^{(\mu+M_j)}$. To illustrate how quickly the complexity can grow, consider the example of Fig. 5.1, with $K = 2$ tracks of interest and $N = 2$ readers. Suppose that the channel response in the absence of frequency offset ($\Delta T_1 = \Delta T_2 = 0$) is:

$$\mathbf{H}(D) = \begin{bmatrix} 1 + 0.5D & 0.4 + 0.16D \\ 0.4 + 0.16D & 1 + 0.5D \end{bmatrix}, \quad (5.12)$$

where $H^{(i,j)}(D)$ denotes the response at reader i from track j . The responses from both tracks have memory $\mu = 1$. Hence, if there were no timing offsets, the standard joint Viterbi algorithm would require 4 states. Suppose instead that $\Delta T_1/T = 2 \times 10^{-5}$ and $\Delta T_2/T = 2 \times 10^{-4}$, and that the length of the sector is $L = 10^4$ bits. Therefore, by the end of the sector, the responses of tracks 1 and 2 will shift by 0.2 bit periods and 2 bit periods, respectively. In order to capture these movements and also to include the significant taps of the moving responses, a reasonable choice for the extra memories would be $M_1 = 2$ and $M_2 = 4$, which would result in a total of $2^3 \times 2^5 = 256$ states. Consequently, compared with the case where both tracks are synchronously written, implementing the Viterbi detector for this example increases the number of states from 4 to 256. Fortunately, as described below, a more efficient implementation of the time-varying target can achieve the same performance with a significant reduction in the amount of extra memory required.

2) Numerical Results

We examine the alternative strategy in detecting $K = 2$ tracks from $N = 2$ readback waveforms of (5.10), where the target and the frequency offset parameters are set according to the example above. We implement a joint Viterbi algorithm where the expected outputs for every transition (p, q) is computed anew for every time k , using (5.11).

The BER performance of the alternative strategy is shown in Fig. 5.5. The figure plots the average BEER for the two tracks being detected, as a function of SNR. (Not shown are the individual error rates for each track, which are a close match to the average because of

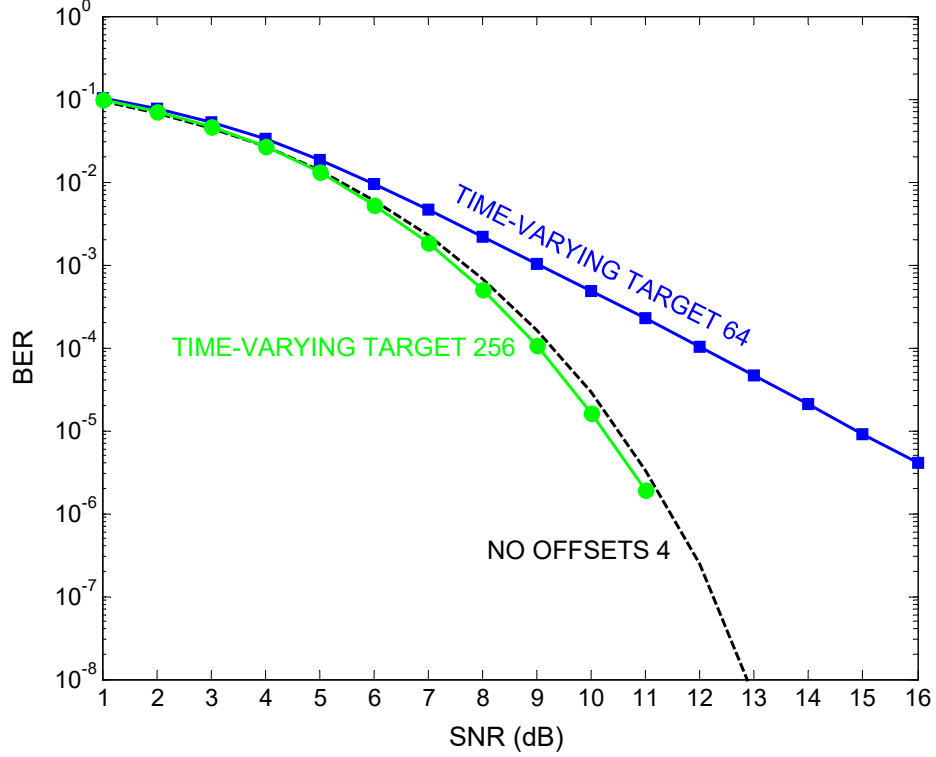


Figure 5.5: BER performance of the alternative strategy with time-varying target based on (5.12) in detecting $K = 2$ tracks from $N = 2$ readback waveform.

the symmetry in this example.)

The curve labeled with “TIME-VARYING TARGET 64” corresponds to the case where $M_1 = M_2 = 2$, yielding $2^3 \times 2^3 = 64$ states. Since at the end of the sector the response of track 1 will shift by 2 bits, the choice of $M_2 = 2$ which adds 2 extra memory to each side of the response, is not big enough to entirely capture this shift. This justifies the poor performance of the 64-state joint Viterbi.

The curve labeled “TIME-VARYING TARGET 256” corresponds to the case where $M_1 = 2$ and $M_2 = 4$, yielding $2^3 \times 2^5 = 256$ states. We observe that, as expected, this choice for the extra memory parameters is sufficient since the performance of the 256-state joint Viterbi closely matches the performance of a standard Viterbi with $2^{2^{\mu=1}} = 4$ states that detects the synchronous system with no timing offsets.

Overall, the only problem seems to be the high complexity due to the extra memory required to capture the moving responses throughout the sector.

3) The ROTAR Algorithm

We have seen how frequency offset causes the time-varying impulse response to both shift and elongate as time progresses. Nevertheless, we can avoid the need for large values of the extra memory parameters $\{M_j\}$, even for extreme cases of large frequency offsets and large sector lengths, if we modify the detector to dynamically track only the significant coefficients of the time-varying impulse response. Towards that objective, let us decompose the k -th timing offset for track j into its integer and fractional parts:

$$\tau_k^{(j)} = d_k^{(j)}T + \theta_k^{(j)}, \quad (5.13)$$

where

$$d_k^{(j)} = \left\lfloor \tau_k^{(j)} / T \right\rfloor \in \{0, 1, 2, \dots\}$$

is the integer part, and

$$\theta_k^{(j)} = \tau_k^{(j)} - d_k^{(j)}T \in [0, T)$$

is the fractional part. With this definition, we can approximate (5.3) as:

$$\begin{aligned} \mathbf{r}_k &= \sum_{j=1}^K \sum_n a_n^{(j)} \mathbf{h}_j(kT - nT - \tau_n^{(j)}) + \mathbf{n}_k \\ &\approx \sum_{j=1}^K \sum_n a_n^{(j)} \mathbf{h}_j(kT - nT - \tau_k^{(j)}) + \mathbf{n}_k \\ &= \sum_{j=1}^K \sum_n a_n^{(j)} \mathbf{h}_j(kT - nT - d_k^{(j)}T - \theta_k^{(j)}) + \mathbf{n}_k \\ &\approx \sum_{j=1}^K \sum_{\ell=-M_j/2}^{\mu+M_j/2} a_{k-\ell-d_k^{(j)}}^{(j)} \mathbf{h}_j(\ell T - \theta_k^{(j)}) + \mathbf{n}_k. \end{aligned} \quad (5.14)$$

The two approximations above are exactly the same as the two used in (5.7) and (5.8) for the isolated track case: The first approximation is valid when the timing offsets are

approximately constant over the duration of the bit response, and the second approximation is valid for sufficiently large parameters $\{M_j\}$.

The model in (5.14) captures the essence of the proposed ROTAR algorithm. Here, the timing offset $\tau_k^{(j)}$ is distributed among the two sides of the convolution: The integer part acts as a *time-varying delay* on the bits, while it is only the fractional part that shifts the responses. The term *rotating target* is derived from the behavior of the target in (5.14) over the duration of a sector: Each time the target approaches a shift of one full bit, the delay d_k increments by one, and the response “rotates” or reverts back to its original unshifted form.

The model in (5.14) is also a noisy version of the output of a time-varying finite-state machine, and thus the ML detector can be implemented using, for example, the Viterbi algorithm with expected outputs for a transition (p, q) at time k computed as:

$$\mathbf{o}_k(p, q) = \sum_{j=1}^K \sum_{\ell=-M_j/2}^{\mu+M_j/2} a_{k-\ell-d_k^{(j)}}^{(j)}(p, q) \mathbf{h}_j(\ell T - \theta_k^{(j)}). \quad (5.15)$$

Because of the time-varying fractional delay, the expected outputs will vary with time and cannot be precomputed. Furthermore, because of the time-varying integer delay of the bits, the structure of the trellis will change each time the integer delay increments.

The key advantage of the rotating property is that it enables us to use small values of the memory parameters $\{M_j\}$, and thus small overall complexity, without any performance loss. In particular, since the fractional delay parameters $\{\theta_k^{(j)}\}$ are limited to the range $[0, T)$, the vast majority of the signal energy of the delayed bit responses can be captured by choosing either $M_j = 2$ or $M_j = 4$ for the asynchronous tracks, with $M_j = 2$ being a reasonable choice for most applications; the complexity disadvantages of moving to $M_j = 4$ will likely outweigh the marginal performance advantages. Thus, the *rotating target* strategy significantly reduces the extra memory required, independent of both the severity of the frequency offset and the sector length.

The pseudocode of the proposed ROTAR algorithm is shown in Algorithm 3 and 4.

Algorithm 3 calls Algorithm 4 to rearrange the states whenever an increment in the integer offset of any of the input tracks is detected. Algorithm 3 adopts the genie-aided assumption that the timing offsets of all the input tracks are known. (The case of unknown timing offsets that must be estimated is handled later, by Algorithm 5.) The inputs to Algorithm 3 are the ADC outputs, the responses from all K tracks and the original memory μ (assumed to be the same for all tracks) of the responses, the maximum of extra memories of all tracks $M_{max} = \max_j M_j$ that will extend the trellis, and the genie-aided timing offsets. The output is the set of detected bits of all K tracks.

Algorithm 3 begins by setting the initial state to state 0, in line 1 where the partial path metric of state 0 and all other states at time 0, respectively, are set to 0 and ∞ . Also, the survivor path for every state p at time 0 ($S_0(p)$) is initialized with an empty vector in line 2. The algorithm then proceeds to the main loop (line 3 through line 16) which steps through each stage of the trellis. The integer and the fractional offsets are computed for every track j in line 4 and line 5, respectively. Line 6 through line 15 step through all state transitions at stage k . The expected outputs and the transition metrics are computed, respectively in line 7 and line 8, for the two transitions from those states p which lead to the state q . Line 9 checks if the integer offset of any of the tracks is incremented. In case of an increment, the trellis should be rearranged and therefore line 10 calls Algorithm 4 to rearrange the partial path metrics $\{\Phi_k(p)\}$ and the survivor paths $\{S_k(p)\}$ for all states p at time k . Thereafter, the algorithm continues exactly as in the standard Viterbi. The predecessor of state q at time $k + 1$ is computed in line 12. The partial path metric of state q at time $k + 1$ is updated in line 13. Also in line 14, the survivor path of state q is updated by concatenating (denoted as operator $|$) the survivor path of the predecessor of state q with the predecessor of state q . Finally, the estimated bits of all K tracks are extracted from the survivor path that minimizes the path metric at the end of the trellis.

The inputs to Algorithm 4 are the partial path metrics, the survivor paths, and the predecessors of all states p at time k , and also the integer bit delays of all tracks j . The

Algorithm 3: ROTAR with Known Timing

Inputs: ADC outputs $\{\mathbf{r}_k\}$, responses $\{\mathbf{h}_j(t)\}$, μ , M_{max} , $\tau_k^{(j)} \forall k, \forall j$
Output: $\{\hat{\mathbf{a}}_j\}$

- 1 **Init:** $\Phi_0(0) = 0$, $\Phi_0(p) = \infty \forall p \neq 0$
- 2 **Init:** $\mathbf{S}_0(p) = [] \forall p$
- 3 **for** $k = 0$ **to** $L + \mu + M_{max} - 1$ **do**
- 4 $d_k^{(j)} = \lfloor \tau_k^{(j)} / T \rfloor \forall j$
- 5 $\theta_k^{(j)} = \tau_k^{(j)} - d_k^{(j)} T \forall j$
- 6 **for** $q = 0$ **to** $Q - 1$ **do**
- 7 Compute $\mathbf{o}_k(p, q)$ using (5.15) $\forall p \rightarrow q$
- 8 $\gamma_k(p, q) = \|\mathbf{r}_k - \mathbf{o}_k(p, q)\|^2 \forall p \rightarrow q$
- 9 **if** $d_k^{(j)} \neq d_{k-1}^{(j)} \forall j$ **then**
- 10 $(\{\Phi_k(p)\}, \{\mathbf{S}_k(p)\})$
 = **Rearrange States** $(\{\Phi_k(p)\}, \{\pi_k(p)\}, \{\mathbf{S}_k(p)\}, \{d_k^{(j)}\})$
- 11 **end**
- 12 $\pi_{k+1}(q) = \underset{p}{\operatorname{argmin}} \{\Phi_k(p) + \gamma_k(p, q)\}$
- 13 $\Phi_{k+1}(q) = \Phi_k(\pi_{k+1}(q)) + \gamma_k(\pi_{k+1}(q), q)$
- 14 $\mathbf{S}_{k+1}(q) = [\mathbf{S}_k(\pi_{k+1}(q)) | \pi_{k+1}(q)]$
- 15 **end**
- 16 **end**
- 17 Extract $\{\hat{\mathbf{a}}_j\}$ from the survivor path that minimizes $\Phi_{L+\mu+M_{max}}$

outputs are the rearranged partial path metrics, and the rearranged survivor paths of all states p at time k . The algorithm begins by declaring an empty vector *collectnewstates* in line 1. From line 2 through line 15, the algorithm finds a new state for every old state and stores it in the vector *collectnewstates*, as follows: First, in line 3, the function **de2bin** converts the decimal *oldstate* to its corresponding binary (over alphabet $\{-1, +1\}$) vector denoted as *OLDSTATE*, so that $OLDSTATE = [OLDSTATE^{(1)}, \dots, OLDSTATE^{(K)}]$, where $OLDSTATE^{(j)}$ denotes the part of the binary vector *OLDSTATE* which corresponds to track j . Likewise in line 4, the predecessor of the *oldstate* is converted to a binary vector denoted as $\Pi = [\Pi^{(1)}, \dots, \Pi^{(K)}]$ where $\Pi^{(j)}$ denotes the part of the predecessor corresponding to track j . An empty vector *newstate* is declared in line 5. From line 6 through line 13, the algorithm steps through each track: In line 7, the track with an increment in its integer

Algorithm 4: Rearrange States

Inputs: $\{\Phi_k(p)\}, \{\mathbf{S}_k(p)\}, \{\pi_k(p)\}, \{d_k^{(j)}\}$
Output: $\{\Phi_k(p)\}, \{\mathbf{S}_k(p)\}$

```
1  collectnewstates = []
2  for oldstate = 0 to  $Q - 1$  do
3      OLDSTATE = de2bin (oldstate)
4       $\Pi$  = de2bin ( $\pi_k(\text{oldstate})$ )
5      newstate = []
6      for  $j = 1$  to  $K$  do
7          if  $d_k^{(j)} \neq d_{k-1}^{(j)}$  then
8               $NEWSTATE^{(j)} = \Pi^{(j)}$ 
9          else
10              $NEWSTATE^{(j)} = OLDSTATE^{(j)}$ 
11          end
12          newstate = [newstate |  $NEWSTATE^{(j)}$ ]
13      end
14      collectnewstates = [collectnewstates | bin2dec (newstate)]
15  end
16   $\Phi_{im}(p) = \Phi_k(p) \forall p$ 
17   $\mathbf{S}_{im}(p) = \mathbf{S}_k(p) \forall p$ 
18  for newstate = 0 to  $Q - 1$  do
19      ind = Find (collectnewstates(newstate) = collectnewstates)
20       $\Phi_k(\text{newstate}) = \min(\Phi_{im}(\text{ind}))$ 
21       $\mathbf{S}_k(\text{newstate}) = \mathbf{S}_{im}(\underset{ind \in \text{ind}}{\text{argmin}}(\Phi_{im}(\text{ind})))$ 
22  end
23   $\Phi_k(p) = \infty \forall p \notin \text{collectnewstates}$ 
```

offset is detected. In case of an increment, the part of the predecessor corresponding to track j should replace the part of the new state corresponding to track j . Hence $NEWSTATE^{(j)}$ is set to $\Pi^{(j)}(p)$, in line 8. Otherwise, the part of the new state corresponding to track j should be the same as the part of the old state corresponding to track j . Hence, in line 10, $NEWSTATE^{(j)}$ is set to $OLDSTATE^{(j)}$. In line 12, the binary vector *newstate* collects every part of the new state corresponding to every track, one by one. In line 14, this vector is converted to decimal to represent the new state for the *oldstate*. Here, *collectnewstates* collects all new states for all old states from 0 to $Q - 1$. The old partial path metrics and the old survivor paths are stored in Φ_{im} and \mathbf{S}_{im} , respectively in line 16 and line 17. Line 18

to line 22 determines the partial path metric and the survivor path for every *newstate*. We should note that, the mapping from every old state to its corresponding new state is not one-to-one. For example, we might find that both states 2 and 6 change to new state 3. In this case, the new state 3 takes over the old state which has the smaller partial path metric. Thus, in line 19, the indices of the duplicate mappings for every *newstate* are found and in line 20, the smaller partial path metric is selected as the partial path metric of the *newstate*. Similarly, in line 21, the survivor path of the state with the smaller partial path metric is selected as the survivor path of the *newstate*. Finally in line 23, those new states that do not appear in the vector *collectnewstates* are killed by setting their partial path metrics to ∞ .

To help appreciate the complexity reduction of the ROTAR algorithm let us revisit the example in Sect. 5.2.2: There we saw that a high-complexity implementation of a joint Viterbi detector with a non-rotating target required the memory parameter values of $M_1 = 2$ and $M_2 = 4$, which results in 256 states. Using ROTAR, however, we can get similar performance using $M_1 = 2$ and $M_2 = 2$, which results in only $2^3 \times 2^3 = 64$ states. Moreover, since the responses from track 2 shift by two bit periods by the end of the sector, we only need to rearrange the structure of the trellis twice through the entire length of the trellis. The computational complexity of this rearrangement in Algorithm 4 is no greater than the computational complexity required to process one stage of the trellis, or equivalently one bit for each track. The extra complexity of Algorithm 2 is thus negligible in relation to the dramatic reduction in overall complexity afforded by the ROTAR algorithm.

4) Locking All ADC's to One Track

The motivation for the ROTAR algorithm stems from the simple observation that it is impossible to simultaneously synchronize the ADC's to multiple asynchronous tracks. Nevertheless, this does not mean that we must resort to using free-running ADC's with sampling rates $1/T$ that are not matched to the bit rate of any of the tracks of interest. Instead, while it is impossible to synchronize the ADC's to multiple tracks simultaneously, we can

always synchronize them to *one* of the tracks, and there is a significant complexity advantage in doing so. Synchronizing all of the ADC's to one track enables us to set the corresponding M_j parameter to zero, significantly reducing the number of trellis states. Synchronizing to one of the tracks can be implemented using a single timing-error detector for the track along with a single PLL that feeds either all of the ADC's (for a real-time implementation) or a bank of interpolative filters, one for each ADC (for a digital implementation).

5) *ROTAR Algorithm with PSP* For the case when the timing offsets are not known, a timing estimation strategy should be used with ROTAR. We propose to use per-survivor processing inside ROTAR to estimate the timings [27]. The algorithm runs a separate PLL for each survivor path, so that every node in the trellis has its own estimate of the timing offsets.

The pseudocode of the proposed ROTAR algorithm with PSP is presented in Algorithm 5. The changes in Algorithm 5 compared to Algorithm 3 are marked with an asterisk. Line 1 implements line 1 and line 2 from Algorithm 3. In line 2, the estimated timing offsets for all states are initialized to zero. Also, in line 3 a variable $sum^{(j)}(p)$ is defined and initialized to zero for every state p and track j . This variable will be used later, in line 15 and line 16, in the PLL update equation. Line 4–line 17 step through each stage of the trellis. In line 5, the maximum estimated integer offset among all states p is selected as the integer offset for all tracks j . This is implemented to help PLL convergence. Line 6 computes the fractional offset for every track j and every state p . Line 7 through line 13 are similar to Algorithm 3, considering that the timings are estimated and are different for each state p . Line 13 implements line 12 – line 14 from Algorithm 3. In line 14, the estimate of the timing offset of every track j is calculated by taking a sum over the estimates which every reader i provides for track j . Here, we have used the Mueller-Muller estimate [23] to compute the error estimate from every reader i for every track j . We should add that the equation in line 14 generalizes the Mueller-Muller estimate, that was originally developed for a 1D

Algorithm 5: ROTAR with PSP

Inputs: ADC outputs $\{\mathbf{r}_k\}$, responses $\{\mathbf{h}_j(t)\}$, μ , M_{max}
Output: $\hat{\mathbf{a}}_j \forall j$
1 Implement line 1–line 2 from Algorithm 3
*2 **Init:** $\hat{\tau}_0^{(j)}(p) = 0 \forall j, \forall p$
*3 **Init:** $sum^{(j)}(p) = 0 \forall j, \forall p$
4 **for** $k = 0$ **to** $L + \mu + M_{max} - 1$ **do**
*5 $\hat{d}_k^{(j)} = \max_p [\hat{\tau}_k^{(j)}(p)/T] \forall j$
*6 $\hat{\theta}_k^{(j)}(p) = \hat{\tau}_k^{(j)}(p) - \hat{d}_k^{(j)}T \forall j, \forall p$
7 **for** $q = 0$ **to** $Q - 1$ **do**
8 Compute $\hat{\mathbf{o}}_k(p, q)$ using (5.15) $\forall p \rightarrow q$
9 $\gamma_k(p, q) = \|\mathbf{r}_k - \hat{\mathbf{o}}_k(p, q)\|^2 \forall p \rightarrow q$
10 **if** $\hat{d}_k^{(j)} \neq \hat{d}_{k-1}^{(j)} \forall j$ **then**
11 $(\{\Phi_k(p)\}, \{\mathbf{S}_k(p)\})$
 $= \text{Rearrange States}(\{\Phi_k(p)\}, \{\pi_k(p)\}, \{\mathbf{S}_k(p)\}, \{\hat{d}_k^{(j)}\})$
12 **end**
13 Implement line 12–line 14 from Algorithm 3
*14 $\hat{\epsilon}_k^{(j)}(q) = \sum_{i=1}^N r_k^{(i)} \hat{o}_{k-1}^{(i,j)}(q) - r_{k-1}^{(i)} \hat{o}_k^{(i,j)}(q) \forall j$
*15 $sum(q) = sum(\pi_{k+1}(q)) + \hat{\epsilon}_{k-1}(\pi_{k+1}(q))$
*16 $\hat{\tau}_{k+1}(q) = \hat{\tau}_k(\pi_{k+1}(q)) + \alpha \hat{\epsilon}_k(q) + \beta sum(q)$
17 **end**
18 **end**
19 Extract $\{\hat{\mathbf{a}}_j\}$ from the survivor path that minimizes $\Phi_{L+\mu+M_{max}}$

setting with only one readback signal, to the MIMO setting with multiple readback signals. Further, line 14 assumes that the N readers significantly cover track j , since otherwise their signal are not useful to provide an estimate for track j . The expected outputs $\{\hat{o}_k^{(i,j)}(q)\}$ are the expected outputs from track j to reader i on the survivor path ending at state q . These outputs are included in the already computed expected outputs of line 8 and are defined according to

$$\hat{o}_k^{(i,j)}(q) = \sum_{\ell=-M_j/2}^{\mu+M_j/2} a_{k-\ell-\hat{d}_k^{(j)}}^{(j)}(q) h_{i,j}(\ell T - \hat{\theta}_k^{(j)}(\pi_{k+1}(q))), \quad (5.16)$$

where $\{a_k^{(j)}(q)\}$ are the bits of track j on the survivor path ending at state q at time $k + 1$. In line 15, the vector variable $\mathbf{sum}(q) = [\mathit{sum}^{(1)}(q), \dots, \mathit{sum}^{(K)}(q)]^T$ sums over the past values of the estimated error vector $\hat{\mathbf{e}}_k = [\hat{e}_k^{(1)}, \dots, \hat{e}_k^{(K)}]^T$ on the survivor path which ends at state q at time $k + 1$. Finally, in line 16, the estimated timing offsets of all K tracks are updated through a second-order PLL.

5.3 Numerical Results

We present performance results of the ROTAR algorithm for the case of $K = 2$ asynchronous tracks with $N = 2$ readers, as illustrated in Fig. 5.1, where the channel model is (5.12). The unknown frequency offset parameters for track 1 and track 2, respectively, are $\Delta T_1/T = 2 \times 10^{-5}$ and $\Delta T_2/T = 2 \times 10^{-4}$. The sector length is $L = 40$ kbits, which results in a maximum slip of 0.8 and 8 bit periods, respectively, for track 1 and track 2 at the end of the sector. The second-order PLL parameters are $\alpha = 0.001$ and $\beta = \alpha^2/4$.

The bit-error rate performance of the proposed ROTAR algorithm with PSP is shown and compared with two other detectors in Fig. 5.6. The figure plots the average of the bit-error probability for the two tracks being detected, as a function of SNR. (Not shown are the individual error rates for each track, which are a close match to the average because of the symmetry in this example.)

The curve labeled “ROTAR 16” shows the performance of a 16-state ROTAR algorithm whose memory parameters are $M_1 = 0$ and $M_2 = 2$. To enable $M_1 = 0$, both ADC’s are locked to track 1 using standard techniques. In particular, prior to detection a standard ITR block consisting of a SISO equalizer, a Viterbi symbol detector, a Mueller-Muller timing-error detector, a second-order PLL, and an interpolation filter plus a secondary interpolation filter were used to lock both readback waveforms to track 1.

The curve labeled “ROTAR 64” shows the performance of a 64-state ROTAR algorithm with memory parameters $M_1 = 2$ and $M_2 = 2$, fed directly by the ADC’s, without any intervening synchronizer that locks to one of the tracks.

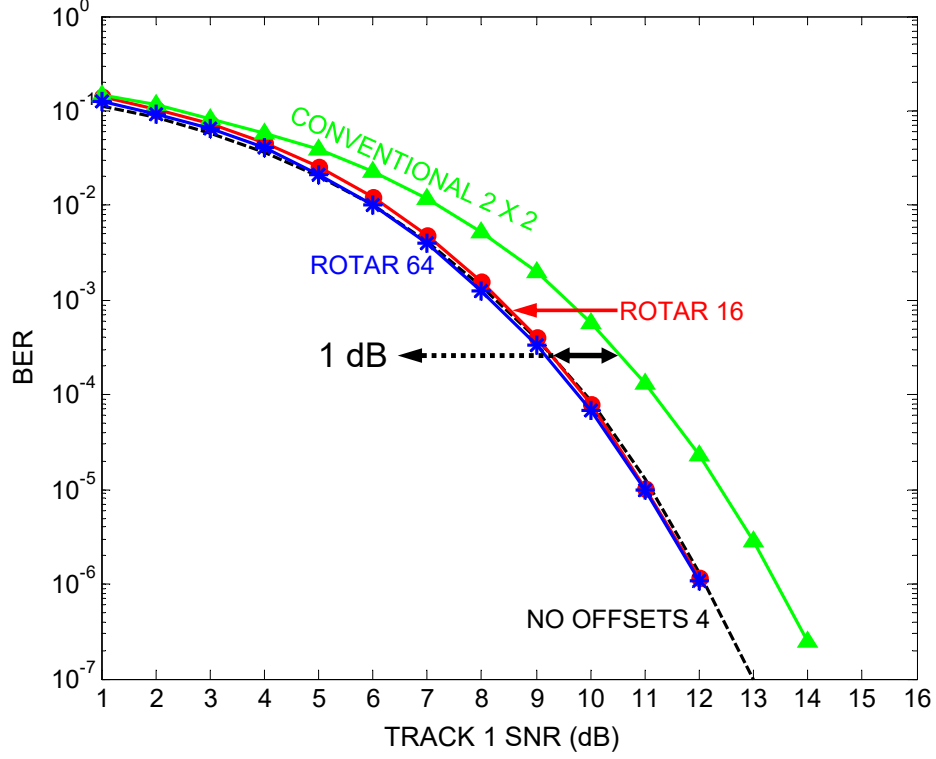


Figure 5.6: BER performance of the ROTAR algorithm with PSP.

The figure shows that the 16-state and 64-state ROTAR detectors have nearly identical performance. Considering the significant reduction in complexity, the advantage of locking to one track is clear. Also shown in Fig. 5.6 is the performance of a conventional receiver that separately detects the two tracks of interest using a pair of independent two-input single-output (MISO) equalizers followed by a pair of independent one-dimensional, 2-state Viterbi detectors with PSP. The performance using this conventional approach is represented by the curve to the right with the triangle markers. We observe that ROTAR outperforms the conventional approach by 1 dB. This performance gain is due to the fact that, in the presence of ITI, joint detection is superior to one-dimensional detection.

Lastly, we also show in Fig. 5.6 the performance of a fictitious system for which the two tracks were written synchronously with each other and also with the ADC sampling rate. The performance of the 4-state joint Viterbi detector for this synchronous case is represented by the dashed lines. Both of the ROTAR detectors are seen to closely match the performance

of the synchronous system, despite the presence of frequency offsets.

5.4 Summary

We proposed the rotating target (ROTAR) algorithm for jointly detecting multiple asynchronous tracks from multiple readback waveforms. ROTAR applies the Viterbi algorithm to a time-varying rotating target that accounts for the asynchrony of the different tracks being detected. To keep complexity low, the timing offsets are decomposed into their integer and fractional parts, and only the fractional parts are used to rotate the target. A further reduction in complexity is realized by locking the ADC's to one track. For the case of unknown timing offsets, ROTAR can use per-survivor processing to embed synchronization inside the joint Viterbi detector.

CHAPTER 6

EQUALIZATION TO A TIME-VARYING TARGET

In the previous chapter, we considered the problem of jointly detecting K asynchronous tracks from N readback waveforms. We proposed the ROTAR algorithm that was based on a joint Viterbi algorithm and a time-varying target. Since we assumed a *perfectly equalized* partial response channel, the time-varying target was essentially the same as the partial response channel itself. In practice, however, an unknown and potentially very long channel response, first, needs to be *equalized* to a short partial response channel before an efficient detection algorithm can be implemented. In a conventional read channel that does single-track detection, a GPR equalization follows timing recovery and precedes detection. For multitrack detection of asynchronous tracks, however, as we explained in the previous chapter, this modular design does no longer work. Instead, the tasks of synchronization and detection should be performed together. Consequently, the GPR equalization should be done first hand, before timing recovery and right after ADC. (This is doable because according to our findings in chapter 4, Sect. 4.2, the solution to the joint optimization of the equalizer filter/s and the target pair is independent of the timing offsets.)

Equalizing before timing recovery has two important implications:

1. since equalization *precedes* timing recovery, the equalized outputs contain a time-varying target. In this sense, the equalized waveforms can be viewed as the outputs of an equalizer that equalizes to a *time-varying target*, and
2. unlike standard GPR equalization (Appendices C and D) where the equalizer and the target pair are computed using synchronous ADC samples, here, they should be computed using asynchronous ADC samples.

In this chapter we propose a GPR equalization strategy that works on asynchronous

ADC outputs and is computed using asynchronous ADC outputs. The proposed equalization strategy is designed to work with the ROTAR detector. At the end of the chapter, we evaluate the performance of the proposed complete read channel that consists of the proposed equalizer and the ROTAR with PSP on a semi realistic data set.

6.1 Channel Model and Assumptions

We consider the same channel model of chapter 5, Sect. 5.1, except that, here, we drop the assumption of a fully equalized channel and instead we consider a full response channel with K input tracks and N output readback waveforms, so that the readback waveform at reader i is:

$$r_i(t) = \sum_{j=1}^K \sum_n a_n^{(j)} f_{i,j}(t - nT - \tau_n^{(j)}) + n_i(t), \quad (6.1)$$

where all the parameters are the same as defined in chapter 5, Sect. 5.1, except for $f_{i,j}(t)$ that is the bit response at reader i from track j , assumed to be bandlimited to half the bit rate.

The i -th readback waveform is filtered by a low-pass antialiasing filter and then sampled at the ADC rate $1/T$, yielding

$$r_k^{(i)} = \sum_{j=1}^K \sum_n a_n^{(j)} f_{i,j}(kT - nT - \tau_n^{(j)}) + n_k^{(i)}. \quad (6.2)$$

Collecting the N samples from each of the N read heads at time k into the vector $\mathbf{r}_k[r_k^{(1)}, \dots, r_k^{(N)}]^T$, and using (5.2), we arrive at a MIMO model:

$$\mathbf{r}_k = \sum_{j=1}^K \sum_n a_n^{(j)} \mathbf{f}_j(kT - nT - \tau_n^{(j)}) + \mathbf{n}_k, \quad (6.3)$$

where $\mathbf{f}_j(t) = [f_{1,j}(t), f_{2,j}(t), \dots, f_{N,j}(t)]^T$ is the vector-valued bit response of track j across all N readers.

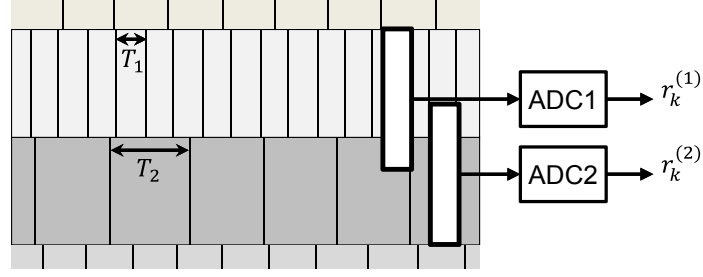


Figure 6.1: An example of two tracks of interest whose timing differ in frequency and phase, and two readers with significant overlap.

6.2 GPR Equalization Strategies to a Time-Varying Target

6.2.1 Prior to Single-Track Detection

Before attacking the problem of equalizing N readback waveforms to a matrix-valued, time-varying target, we first consider the simpler problem of equalizing N readback waveforms to a vector-valued, time-varying target, prior to single-track detection. In the following, we describe three strategies for equalizing the vector of N readback waveforms of (6.3) so that a following ML detector can detect a single track.

The three equalization strategies are illustrated in Fig. 6.2. Each strategy includes a lower branch that is only used for computing the unknown equalizer and the target pair during the preamble of the sector where the user bits are known. The resulted equalizer and target pair are used during the remaining of the sector for the detection of the unknown bits.

1) The Conventional Strategy

The conventional strategy is to separately synchronize each readback waveform before equalization. This strategy for the example of Fig. 6.1 is shown in Fig. 6.2a where track 2 is to be detected.

In general, after synchronizing N waveforms to the timings of track t , the vector of

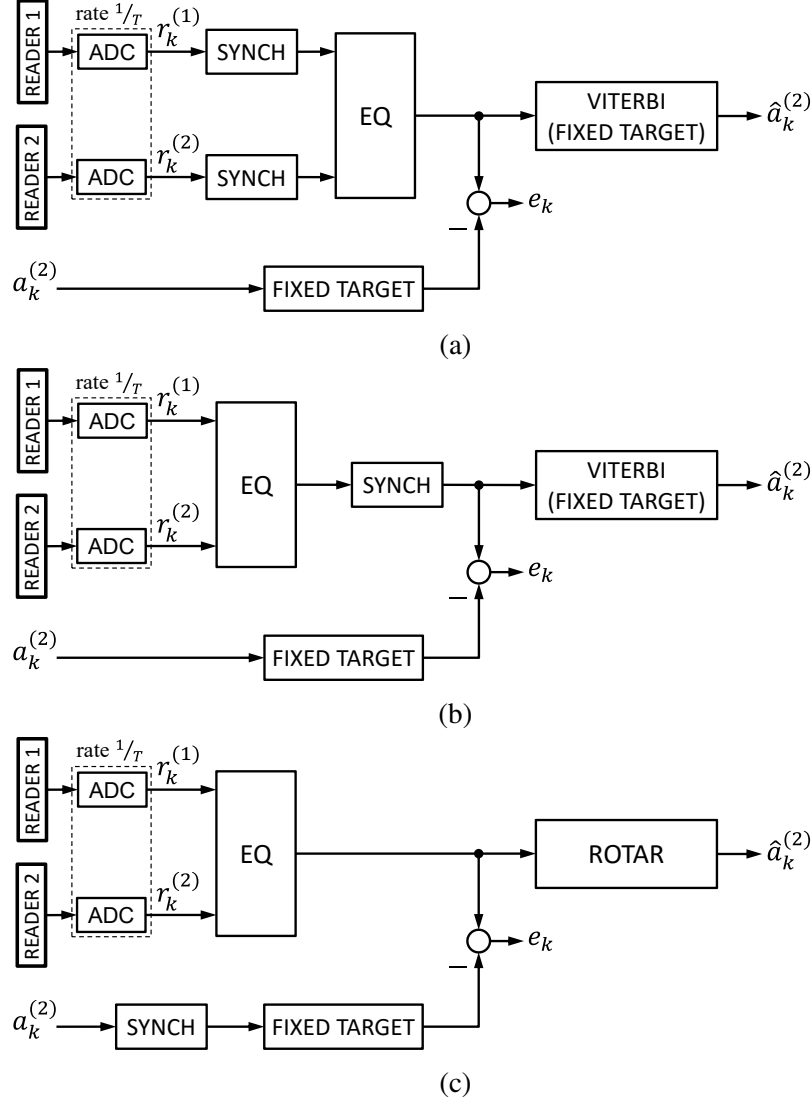


Figure 6.2: GPR equalization strategies, for the exampling given in Fig. 6.1 and detecting track 2: (a) conventional strategy, (b) alternative strategy 1, and (c) alternative strategy 2.

equalized waveforms is given as:

$$\begin{aligned}
 \mathbf{r}_k &= \sum_n a_n^{(t)} \mathbf{f}_t(kT - nT - \tau_n^{(t)} + \tau_n^{(t)}) + \sum_{j \neq t} \sum_n a_n^{(t)} \mathbf{f}_j(kT - nT - \tau_n^{(j)} + \tau_n^{(t)}) + \mathbf{n}_k, \\
 &= \sum_n a_n^{(t)} \mathbf{f}_t(kT - nT) + \sum_{j \neq t} \sum_n a_n^{(2)} \mathbf{f}_j(kT - nT - \tau_n^{(j)} + \tau_n^{(t)}) + \mathbf{n}_k, \\
 &= \sum_n a_n^{(t)} \mathbf{f}_t(kT - nT) + \boldsymbol{\eta}_k,
 \end{aligned} \tag{6.4}$$

where $\boldsymbol{\eta}_k$ is the sum of the interference from other tracks $j \neq t$ and the AWGN noise.

Equation (6.4) is the output of a channel with no timing offsets and therefore, it can be equalized using a standard GPR equalization technique. The MMSE solution for jointly computing the N equalizer filters and the fixed target in Fig. 6.2a is given in Appendix C. After equalization to a fixed target, a standard Viterbi algorithm, for example, detects track t . The same exact process can then be repeated for each of the remaining tracks.

2) *Alternative Strategy 1*

Alternatively, according to the result of chapter 4, Sect. 4.2, we can delay synchronization until after equalization. The immediate advantage is reducing the number of synchronization blocks from N to only K synchronization blocks. This strategy for the example of detecting track 2 from $N = 2$ waveforms is shown in Fig. 6.2b.

Since for practical timing offsets, the equalizer and target pair are transparent to the timing offsets, the equalizer and target pair in Fig. 6.2b is essentially the same the ones in Fig. 6.2a. Nevertheless, the computation of these variables does no longer follow the standard technique described in Appendix C, and is different in two aspects: 1) the mean-squared error should be minimized after the synchronizer block, and 2) the unknown timing offsets of track j $\{\tau_n^{(j)}\}$ should also be optimized along with the equalizer and the target pair. In the following, we outline a solution for joint optimization of the equalizer filters, the target, and the timing offsets $\{\tau_n^{(j)}\}$, where the final objective is to detect track j .

Fig. 6.3 illustrates a parametric analogy of the equalization strategy of Fig. 6.2b. During training, the user bits on a track of interest, j , are known, however, the timing offsets are unknown and should be estimated. Therefore, the problem is to jointly optimize for the equalizer filters, the target, and the timing offsets. Let $r_\ell^{(i)}$ denote the ℓ -th sample of the i -th readback waveform for $i \in \{1, \dots, N\}$ sampled at rate $1/T$. Combining the N samples at time ℓ into a $N \times 1$ vector yields:

$$\mathbf{r}_\ell = \begin{bmatrix} r_\ell^{(1)} \\ \vdots \\ r_\ell^{(N)} \end{bmatrix} \quad (6.5)$$

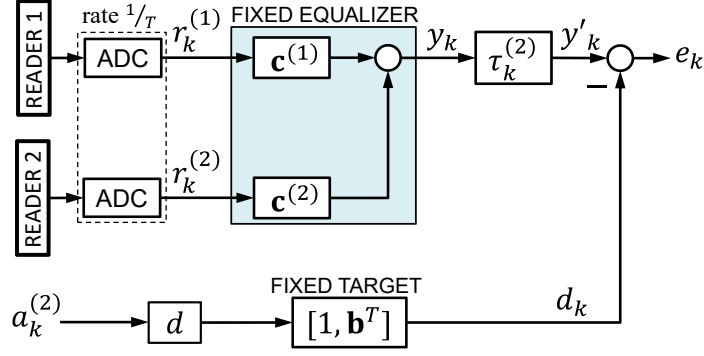


Figure 6.3: A parametric illustration of the alternative equalization strategy 1 for the case of detecting a single track from $N = 2$ waveforms as shown in Fig. 6.2b.

We define each equalizer filter i with Nc coefficients, according to $\mathbf{c}^{(i)} = [c_0^{(i)}, c_1^{(i)}, \dots, c_{Nc-1}^{(i)}]^T$. Combining the N coefficients across N filters at time ℓ into a $N \times 1$ vector also yields:

$$\mathbf{c}_\ell = \begin{bmatrix} c_\ell^{(1)} \\ \vdots \\ c_\ell^{(N)} \end{bmatrix}. \quad (6.6)$$

The N equalizer filter outputs are added so that the bank of N equalizer filters can be viewed as an N -input single-output equalizer with coefficients $\{\mathbf{c}_0, \dots, \mathbf{c}_{Nc-1}\}$, whose k -th output can be written as $y_k = \underline{\mathbf{c}}^T \underline{\mathbf{r}}_k$, where

$$\underline{\mathbf{c}} = \begin{bmatrix} \mathbf{c}_0 \\ \vdots \\ \mathbf{c}_{Nc-1} \end{bmatrix}_{NNc \times 1} \quad \text{and} \quad \underline{\mathbf{r}}_k = \begin{bmatrix} \mathbf{r}_k \\ \vdots \\ \mathbf{r}_{k-Nc+1} \end{bmatrix}_{NNc \times 1}. \quad (6.7)$$

The equalizer outputs $\{y_k\}$, then, should be synchronized to the timings of track j , via interpolation, so that the signals from both branches are synchronous to each other. Since we consider a frequency offset model where $\tau_k^{(j)} = k\Delta T_j$, an interpolation filter such as

$$\mathbf{s}_k = [s((l - k\Delta T_j)T), \dots, s((-l - k\Delta T_j)T)]^T, \quad (6.8)$$

where $s(t) = \frac{\sin(\pi t/T)}{(\pi t/T)}$, and l is the delay of the filter, outputs

$$y'_k = \mathbf{s}_k^T \mathbf{y}_k, \quad (6.9)$$

where

$$\mathbf{y}_k = \begin{bmatrix} y_k \\ \vdots \\ y_{k-Ns+1} \end{bmatrix}_{Ns \times 1}, \quad (6.10)$$

and where N_s is the length of the interpolation filter.

We consider the target to be monic and of the form $[1, \mathbf{b}^T]$, where $\mathbf{b} = [b_1, \dots, b_\mu]^T$ is the target tail and μ is the target memory. Filtering the bits on track j by the target yields the signal $a_{k-d}^{(j)} + \mathbf{b}^T \mathbf{a}_k^{(j)}$, where d is the delay parameter of the equalizer and the interpolation filter cascade, and where $\mathbf{a}_k^{(j)} = [a_{k-d-1}^{(j)}, \dots, a_{k-d-\mu}^{(j)}]^T$. With this terminology, the optimization problem is to jointly choose the equalizer $\underline{\mathbf{c}}$ and the target \mathbf{b} to minimize the mean-squared error $MSE = E(e_k^2)$ where

$$e_k = \mathbf{s}_k^T \mathbf{y}_k - a_{k-d}^{(j)} - \mathbf{b}^T \mathbf{a}_k^{(j)}. \quad (6.11)$$

Since the error includes a time varying interpolation filter \mathbf{s}_k , a closed form MMSE solution can not be given. Rather, an adaptive solution with training such as the least mean squares (LMS) algorithm can be implemented. According to LMS, the equalizer filters and the target tail are updated as:

$$\begin{aligned} \underline{\mathbf{c}} &= \underline{\mathbf{c}} - \nu e_k \underline{\mathbf{r}}_k \\ \mathbf{b} &= \mathbf{b} + \nu e_k \mathbf{a}_k^{(j)}, \end{aligned} \quad (6.12)$$

where ν is the LMS step size. To simultaneously adapt the interpolation filter \mathbf{s}_k , at each iteration of the LMS, we only need to update an estimate of $\tau_k^{(j)} = k\Delta T_j$. Therefore, it is

convenient to use a second-order PLL for this purpose, so that:

$$\hat{\tau}_{k+1}^{(j)} = \hat{\tau}_k^{(j)} + \alpha \hat{\epsilon}_k + \beta \sum_{\ell=1}^{k-1} \hat{\epsilon}_\ell, \quad (6.13)$$

where $\hat{\epsilon}_k$ is the timing error estimate given by the M&M TED equation:

$$\hat{\epsilon}_k = y'_k d_{k-1} - y'_{k-1} d_k, \quad (6.14)$$

where

$$d_k = \sum_{\ell=0}^{\mu} h_\ell a_{k-\ell-d}^{(j)} \quad (6.15)$$

is the ideal, noiseless, equalized ADC output at time k , and $\mathbf{h} = [h_0, \dots, h_\mu] = [1, \mathbf{b}^T]$ is the target.

Equations (6.12) and (6.13), respectively, update the equalizer and the target pair, and the interpolation filter $\mathbf{s}_k = [s((l - \hat{\tau}_k^{(j)})T), \dots, s((-l - \hat{\tau}_k^{(j)})T)]^T$.

2) Alternative Strategy 2

A second alternative strategy is to further delay the synchronization until inside the final detection, and jointly synchronize and detect using the ROTAR algorithm. Fig. 6.2c illustrates the equalization strategy that precedes the ROTAR algorithm. In computing the equalizer and the target pair, the user bits on the track of interest should be *delayed* prior to being filtered by the target, so that the signals from both branches are synchronized. A parametric representation of Fig. 6.2c is shown in Fig. 6.4. An interpolation filter, here, should *delay* the bits, and therefore a sinc interpolation, for example, should be of the form

$$\mathbf{s}_k = [s((l + k\Delta T_j)T), \dots, s((-l + k\Delta T_j)T)]^T. \quad (6.16)$$

Thus, the delayed bits on track j are given as:

$$a_k'^{(j)} = \mathbf{s}_k^T \mathbf{a}_k^{(j)}, \quad (6.17)$$

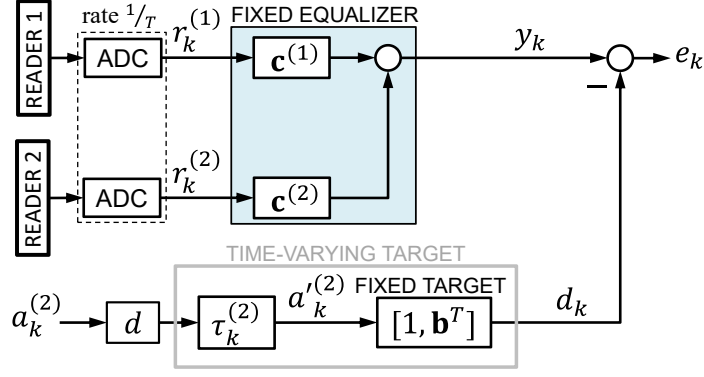


Figure 6.4: A parametric illustration of the alternative equalization strategy 2 for the case of detecting a single track from $N = 2$ waveforms as shown in Fig. 6.2c.

where $\mathbf{a}_k^{(j)} = [a_{k-d}^{(j)}, \dots, a_{k-d-Ns+1}^{(j)}]^T$ and where $d + l$ is the delay of the equalizer cascade.

With this terminology, the error is written as:

$$e_k = \underline{\mathbf{c}}^T \underline{\mathbf{r}}_k - a_k'^{(j)} - \mathbf{b}^T \mathbf{a}_k'^{(j)}, \quad (6.18)$$

where $\mathbf{a}_k'^{(j)} = [a_{k-1}'^{(j)}, \dots, a_{k-\mu}'^{(j)}]^T$. Similar to alternative strategy 1, it is convenient to employ an LMS solution for updating the equalizer and target pair, according to:

$$\begin{aligned} \underline{\mathbf{c}} &= \underline{\mathbf{c}} - \nu e_k \underline{\mathbf{r}}_k \\ \mathbf{b} &= \mathbf{b} + \nu e_k \mathbf{a}_k'^{(j)}, \end{aligned} \quad (6.19)$$

along with a second-order PLL for estimating the timing offset $\tau_k^{(j)}$ according to (6.13), where the M&M TED equation follows

$$\hat{e}_k = y_k d_{k-1} - y_{k-1} d_k, \quad (6.20)$$

and

$$d_k = \sum_{\ell=0}^{\mu} h_{\ell} a_{k-\ell}'^{(j)}. \quad (6.21)$$

It is interesting to note that the target and the interpolation cascade in Fig. 6.4, together

can be viewed as a *time-varying target*, and in turn, the *fixed equalizer* can be viewed as an equalizer that equalizes to a time-varying target as a result.

The optimum solution for the equalizer and the target pair in Fig. 6.4 is essentially the same as in Fig. 6.3 for alternative strategy 1, and also the same as in Fig. 6.2a for the conventional GPR. In other words, the optimum equalizer and target pair is essentially the same throughout all three strategies. Nevertheless, it is the computation of the solution that differs among them.

6.2.2 Prior to Multitrack Detection

Let us turn our focus back to the multitrack detection of K tracks from N readback waveforms. Unlike the single-track detection of the previous section, here the synchronization and detection must be performed jointly. Therefore, the conventional GPR strategy where each waveform is separately synchronized (prior to the equalization and detection) to a track of interest, can no longer work. Neither does the alternative strategy 1, since in the alternative strategy 1, similar to the conventional GPR strategy, the equalized waveform can only be synchronized to the timings of only *one* track. Nevertheless, the multitrack extension of the alternative strategy 2 based on a time-varying target perfectly suits our purpose. Fig. 6.5 illustrates the alternative strategy 2 for the example of $K = 2$ asynchronous tracks and $N = 2$ overlapping readers of Fig. 6.1.

In general, there are at least $K \times N$ individual equalizer filters, the bank of which constructs a N -input K -output MIMO equalizer. We define each equalizer filter $j, i \in \{1, \dots, K\}, \{1, \dots, N\}$ with N_c coefficients, according to $\mathbf{c}_{j,i} = [c_0^{(j,i)}, c_1^{(j,i)}, \dots, c_{N_c-1}^{(j,i)}]^T$. The MIMO equalizer can be viewed as the bank of K individual N -input single-output MISO equalizers. Combining the N coefficients of the j -th MISO equalizer at time ℓ into a

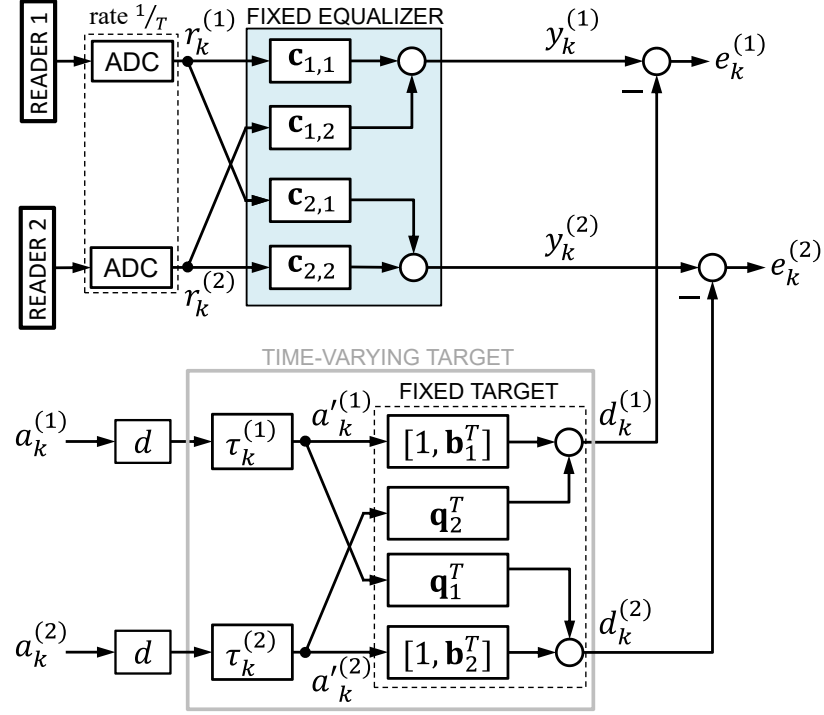


Figure 6.5: A parametric illustration of the alternative strategy 2 for the case of jointly detecting $K = 2$ tracks from $N = 2$ waveforms as shown in Fig. 6.1.

$N \times 1$ vector yields:

$$\mathbf{c}_\ell^{(j)} = \begin{bmatrix} c_\ell^{(j,1)} \\ \vdots \\ c_\ell^{(j,N)} \end{bmatrix}. \quad (6.22)$$

Thus, the output of the j -th MISO equalizer is $y_k^{(j)} = \underline{\mathbf{c}}_j^T \underline{\mathbf{r}}_k$, where

$$\underline{\mathbf{c}}_j = \begin{bmatrix} \mathbf{c}_0^{(j)} \\ \vdots \\ \mathbf{c}_{Nc-1}^{(j)} \end{bmatrix}_{NNc \times 1} \quad \text{and} \quad \underline{\mathbf{r}}_k = \begin{bmatrix} \mathbf{r}_k \\ \vdots \\ \mathbf{r}_{k-Nc+1} \end{bmatrix}_{NNc \times 1}. \quad (6.23)$$

Each MISO equalizer can be paired with a matrix-valued target. We consider each target

j to have the form

$$H_j = \begin{bmatrix} 1 & \mathbf{b}_j^T \\ & \mathbf{q}_{\ell_1}^T \\ & \vdots \\ & \mathbf{q}_{\ell_{K-1}}^T \end{bmatrix}_{K \times (\mu+1)}, [\ell_1, \dots, \ell_{K-1}] = [1, \dots, j-1, j+1, \dots, K], \quad (6.24)$$

where the first row includes $\mathbf{b}_j = [b_1^{(j)}, \dots, b_\mu^{(j)}]^T$, the response tail from track j , and where the rest of the rows are the responses $\{\mathbf{q}_{\ell_m}\}$ from other $K-1$ tracks. Since each output signal from the target branch should be synchronized to the corresponding output signal from the equalizer branch, the bits on each track j should be delayed by $\{\tau_k^{(j)}\}$ using an interpolation filter such as the sinc interpolation filter given in (6.16), according to:

$$a'_k{}^{(j)} = \mathbf{s}_k^{(j)T} \mathbf{a}_k^{(j)}. \quad (6.25)$$

Filtering the delayed user bits on all K tracks with the j -th target yields:

$$d_k^{(j)} = a'_k{}^{(j)} + \mathbf{b}_j^T \mathbf{a}'_k{}^{(j)} + \sum_{m=1}^{K-1} \mathbf{q}_{\ell_m}^T \begin{bmatrix} a'^{(\ell_m)}_k \\ \mathbf{a}'^{(\ell_m)}_k \end{bmatrix}, [\ell_1, \dots, \ell_{K-1}] = [1, \dots, j-1, j+1, \dots, K]. \quad (6.26)$$

The optimization problem is to jointly choose the K MISO equalizers $\{\mathbf{c}_j\}$, the K targets parameters $\{\mathbf{b}_j\}$ and $\{\mathbf{q}_j\}$, and the timings of the K tracks $\{\tau_k^{(j)}\}$ to minimize the mean-squared error

$$MSE = E\left(\sum_{j=1}^K e_k^{(j)2}\right), \quad (6.27)$$

where

$$e_k^{(j)} = \mathbf{c}_j^T \mathbf{r}_k - d_k^{(j)}. \quad (6.28)$$

Similar to single-track detection in the previous section, we employ an adaptive algo-

rithm, such as LMS, to update each MISO equalizer and the corresponding target pair, according to:

$$\begin{aligned}\mathbf{c}_j &= \mathbf{c}_j - \nu e_k^{(j)} \mathbf{r}_k \\ \mathbf{b}_j &= \mathbf{b}_j + \nu e_k^{(j)} \mathbf{a}'_k^{(j)} \\ \mathbf{q}_j &= \mathbf{q}_j + \nu e_k^{(j)} [\mathbf{a}'_k^{(j)}, \mathbf{a}'_k^{(j)T}]^T.\end{aligned}\tag{6.29}$$

Also, a second-order PLL for every track j can update the estimated timings of track j according to:

$$\hat{\tau}_{k+1}^{(j)} = \hat{\tau}_k^{(j)} + \alpha \hat{\epsilon}_k^{(j)} + \beta \sum_{\ell=1}^{k-1} \hat{\epsilon}_\ell^{(j)},\tag{6.30}$$

where the j -th M&M TED output is computed as:

$$\hat{\epsilon}_k^{(j)} = y_k^{(j)} d_{k-1}^{(j)} - y_{k-1}^{(j)} d_k^{(j)}.\tag{6.31}$$

At each iteration of the LMS algorithm, the equations (6.29), (6.30), and (6.31) should be updated for all K tracks.

An alternative to LMS is the recursive least squares (RLS) algorithm that unlike LMS minimizes a *weighted* sum of squared errors, given by

$$WMSE = \sum_{\ell=0}^k \lambda^{k-\ell} \sum_{j=1}^K e_k^{(j)2},\tag{6.32}$$

where $0 < \lambda \leq 1$ is the *forgetting factor* which gives exponentially less weight to older error samples. Compared to LMS, RLS exhibits extremely fast convergence, the benefit that comes at the cost of higher computational complexity. RLS is similar to the closed-form MMSE solution described in Appendices C and D, in that it estimates the vector of unknowns $\mathbf{w}_j = \mathbf{R}_{vv}^{(j)-1} \mathbf{p}_j$, but *recursively*, releasing the need to compute matrix inverse. In the following we derive RLS update equations for the problem of choosing the MIMO

equalizer filters and the target parameters of the alternative strategy2, in order to minimize the weighted mean-squared error of (6.32).

Replacing (6.26) into (6.28), the error corresponding to the j -th MISO equalizer (6.28) can be written as:

$$\begin{aligned} e_k^{(j)} &= \underline{\mathbf{c}}_j^T \underline{\mathbf{r}}_k - a_k'^{(j)} - \mathbf{b}_j^T \mathbf{a}_k'^{(j)} - \sum_{m=1}^{K-1} \mathbf{q}_{\ell_m}^T \begin{bmatrix} a_k'^{(\ell_m)} \\ \mathbf{a}_k'^{(\ell_m)} \end{bmatrix}, [\ell_1, \dots, \ell_{K-1}] = [1, \dots, j-1, j+1, \dots, K] \\ &= \mathbf{w}_j^T \mathbf{v}_k^{(j)} - a_k'^{(j)}, \end{aligned} \quad (6.33)$$

where similar to the MMSE approach

$$\mathbf{w}_j = \begin{bmatrix} \underline{\mathbf{c}}_j \\ -\mathbf{b}_j \\ -\mathbf{q}_{\ell_1} \\ \vdots \\ -\mathbf{q}_{\ell_{K-1}} \end{bmatrix}_{(NNc+K\mu+K-1) \times 1} \quad \text{and} \quad \mathbf{v}_k^{(j)} = \begin{bmatrix} \underline{\mathbf{r}}_k \\ \mathbf{a}_k'^{(j)} \\ \begin{bmatrix} a_{k-d}'^{(\ell_1)} \\ \mathbf{a}_k'^{(\ell_1)} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} a_{k-d}'^{(\ell_{K-1})} \\ \mathbf{a}_k'^{(\ell_{K-1})} \end{bmatrix} \end{bmatrix}_{(NNc+K\mu+K-1) \times 1}. \quad (6.34)$$

In order to minimize (6.32), we choose \mathbf{w}_j such that the derivative of (6.32) with respect to \mathbf{w}_j is zero, according to:

$$\begin{aligned} \frac{\partial W_{MSE}}{\partial \mathbf{w}_j} &= 2 \sum_{\ell=0}^k \lambda^{k-\ell} e_\ell^{(j)} \frac{\partial e_\ell^{(j)}}{\partial \mathbf{w}_j} \\ &= 2 \sum_{\ell=0}^k \lambda^{k-\ell} e_\ell^{(j)} \mathbf{v}_\ell^{(j)} = 0. \end{aligned} \quad (6.35)$$

By substituting (6.28) into above we get:

$$\sum_{\ell=0}^k \lambda^{k-\ell} (\mathbf{v}_\ell^{(j)})^T \mathbf{w}_j - a'_\ell^{(j)} \mathbf{v}_\ell^{(j)} = 0, \quad (6.36)$$

that gives

$$\sum_{\ell=0}^k \lambda^{k-\ell} \mathbf{v}_\ell^{(j)} \mathbf{v}_\ell^{(j)T} \mathbf{w}_j = \sum_{\ell=0}^k \lambda^{k-\ell} a'_\ell^{(j)} \mathbf{v}_\ell^{(j)}, \quad (6.37)$$

and therefore

$$\mathbf{w}_k^{(j)} = \mathbf{R}_{vv_k}^{(j)-1} \mathbf{p}_k^{(j)}, \quad (6.38)$$

where the covariance matrix and the cross-covariance vector are respectively given by

$$\begin{aligned} \mathbf{R}_{vv_k}^{(j)} &= \sum_{\ell=0}^k \lambda^{k-\ell} \mathbf{v}_\ell^{(j)} \mathbf{v}_\ell^{(j)T}, \\ \mathbf{p}_k^{(j)} &= \sum_{\ell=0}^k \lambda^{k-\ell} a'_\ell^{(j)} \mathbf{v}_\ell^{(j)}. \end{aligned} \quad (6.39)$$

Equation (6.38) looks exactly similar to the MMSE solution (D.12) except for the fact that the expected values in the covariance matrix and cross-covariance vector are replaced with a weighted sum. In contrast to MMSE, however, RLS computes (6.38) by recursively computing the covariance matrix and the cross-covariance vector, according to:

$$\begin{aligned} \mathbf{R}_{vv_k}^{(j)} &= \lambda \mathbf{R}_{vv_{k-1}}^{(j)} + \mathbf{v}_k^{(j)} \mathbf{v}_k^{(j)T}, \\ \mathbf{p}_k^{(j)} &= \lambda \mathbf{p}_{k-1}^{(j)} + a'_k^{(j)} \mathbf{v}_k^{(j)}. \end{aligned} \quad (6.40)$$

The inverse of the covariance matrix is therefore given by:

$$\begin{aligned} \mathbf{R}_{vv_k}^{(j)-1} &= \mathbf{P}_k^{(j)} = [\lambda \mathbf{R}_{vv_{k-1}}^{(j)} + \mathbf{v}_k^{(j)} \mathbf{v}_k^{(j)T}]^{-1} \\ &= \lambda^{-1} \mathbf{P}_{k-1}^{(j)} - \lambda^{-1} \mathbf{P}_{k-1}^{(j)} \mathbf{v}_k^{(j)} [1 + \mathbf{v}_k^{(j)T} \lambda^{-1} \mathbf{P}_{k-1}^{(j)} \mathbf{v}_k^{(j)}]^{-1} \mathbf{v}_k^{(j)T} \lambda^{-1} \mathbf{P}_{k-1}^{(j)} \\ &= \lambda^{-1} \mathbf{P}_{k-1}^{(j)} - \mathbf{g}_k^{(j)} \mathbf{v}_k^{(j)T} \lambda^{-1} \mathbf{P}_{k-1}^{(j)}, \end{aligned} \quad (6.41)$$

where in the second line we used the Woodbury matrix identity $(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$, and where the gain vector \mathbf{g}_k in the third line is:

$$\mathbf{g}_k^{(j)} = \lambda^{-1} \mathbf{P}_{k-1}^{(j)} \mathbf{v}_k^{(j)} [1 + \mathbf{v}_k^{(j)T} \lambda^{-1} \mathbf{P}_{k-1}^{(j)} \mathbf{v}_k^{(j)}]^{-1}. \quad (6.42)$$

The gain vector can be written as:

$$\mathbf{g}_k^{(j)} [1 + \mathbf{v}_k^{(j)T} \lambda^{-1} \mathbf{P}_{k-1}^{(j)} \mathbf{v}_k^{(j)}] = \lambda^{-1} \mathbf{P}_{k-1}^{(j)} \mathbf{v}_k^{(j)}, \quad (6.43)$$

that gives:

$$\begin{aligned} \mathbf{g}_k^{(j)} &= \lambda^{-1} [\mathbf{P}_{k-1} - \mathbf{g}_k^{(j)} \mathbf{v}_k^{(j)T} \mathbf{P}_{k-1}] \mathbf{v}_k^{(j)} \\ &= \mathbf{P}_k \mathbf{v}_k^{(j)}, \end{aligned} \quad (6.44)$$

where in the third line we have used (6.41).

Finally, we can derive the recursive equation for \mathbf{w}_j , as follow:

$$\begin{aligned} \mathbf{w}_k^{(j)} &= \mathbf{P}_k^{(j)} \mathbf{p}_k^{(j)} = \lambda \mathbf{P}_{k-1}^{(j)} \mathbf{p}_{k-1}^{(j)} + a_k'^{(j)} \mathbf{P}_k^{(j)} \mathbf{v}_k^{(j)} \\ &= \lambda [\lambda^{-1} \mathbf{P}_{k-1}^{(j)} - \mathbf{g}_k^{(j)} \mathbf{v}_k^{(j)T} \lambda^{-1} \mathbf{P}_{k-1}^{(j)}] \mathbf{p}_{k-1}^{(j)} + a_k'^{(j)} \mathbf{g}_k^{(j)} \\ &= \mathbf{P}_{k-1}^{(j)} \mathbf{p}_{k-1}^{(j)} + \mathbf{g}_k^{(j)} [a_k'^{(j)} - \mathbf{v}_k^{(j)T} \mathbf{P}_{k-1}^{(j)} \mathbf{p}_{k-1}^{(j)}] \\ &= \mathbf{w}_{k-1}^{(j)} - \mathbf{g}_k^{(j)} v_k^{(j)}, \end{aligned} \quad (6.45)$$

where in the second line we have used (6.41) and (6.44), and in the last line we have used (6.38), and where $v_k^{(j)}$ is the *a priori* error, the error before updating the vector \mathbf{w}_j , and is defined as:

$$v_k^{(j)} = \mathbf{v}_k^{(j)T} \mathbf{w}_{k-1}^{(j)} - a_k'^{(j)}. \quad (6.46)$$

Algorithm 6: Alternative strategy 2 with RLS, second-order PLL, and M&M TED

Inputs: ADC outputs $\{\mathbf{r}_k\}$, user bits $\{a_k^{(j)}\} \forall j$, equalizer size N_C , target memory μ , forgetting factor λ , regularization factor δ , and PLL parameters α and β

Output: $\mathbf{w}_j \forall j$

```

1  Init:  $\hat{\tau}_0^{(j)} = 0 \forall j$ 
2  Init:  $\mathbf{w}_0^{(j)} = \mathbf{0} \forall j$ 
3  Init:  $\mathbf{P}_0^{(j)} = \delta^{-1} \mathbf{I}_{NN_C+K\mu+K-1} \forall j$ 
4  for  $k = 1$  to  $L$  do
5      for  $j = 1$  to  $K$  do
6          Compute  $a_k'^{(j)}$  using (6.25) and (6.16)
7          Compute a priori error  $v_k^{(j)}$  using (6.46)
8          Compute the gain vector  $\mathbf{g}_k^{(j)}$  using (6.42)
9          Compute the inverse covariance matrix  $\mathbf{P}_k^{(j)}$  using (6.41)
10         Update the weight vector  $\mathbf{w}_k^{(j)}$  using (6.45)
11         Update  $\hat{\tau}_k^{(j)} = \hat{\tau}_{k-1}^{(j)} + \alpha \hat{\epsilon}_{k-1}^{(i)} + \beta \sum_{\ell=0}^{k-2} \hat{\epsilon}_\ell^{(i)}$  using (6.13)
12     end
13 end

```

Compare this with the *a posteriori* error that results after updating \mathbf{w}_j :

$$e_k^{(j)} = \mathbf{v}_k^{(j)T} \mathbf{w}_k^{(j)} - a_k'^{(j)}. \quad (6.47)$$

We can now outline the RLS algorithm combined with a second-order PLL for updating each of the K MISO equalizers and their corresponding targets following the alternative strategy 2 for GPR equalization. The pseudocode is presented in Fig. 6. The inputs to the algorithm are the asynchronous ADC outputs, the user bits on all K tracks, the forgetting factor, and the regularization factor for initiating the inverse covariance matrix. The outputs are the weight vectors of all MISO equalizers and their corresponding targets. During the preamble of length L , the user bits are known and the algorithm can jointly update the delayed bits, using the timing estimates of a second-order PLL, and the optimum weight vectors $\{\mathbf{w}_j\}$.

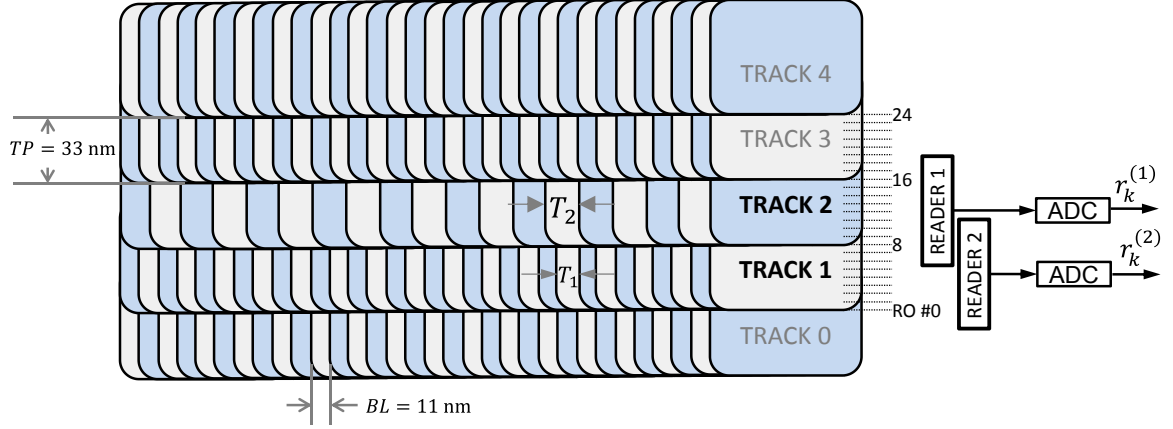


Figure 6.6: The description of DSI data set. 25 different reader positions separated by $TP/8$ are marked with dashed lines. The resulted 25 readback waveforms mainly cover the three middle tracks. The data is generated from the grain-flipping probability model. A write timing frequency offset of $\Delta T_2 = 2 \times 10^{-4}$ is injected into tracks 2 while all other tracks are written synchronously to each other. As shown, we used two reader positions in the simulations to detect tracks 1 and 2.

6.3 Numerical Results

In this section we assess the performance of the proposed alternative strategy 2 of GPR equalization for the case of $K = 2$ asynchronous tracks with $N = 2$ readback waveforms as illustrated in Fig. 6.1. We also provide ultimate BER performance of our proposed read channel that includes the alternative strategy 2 of GPR equalization and the ROTAR algorithm of the previous chapter for joint detection of the two tracks. The simulations are performed on a data set provided by data storage institute (DSI). The waveforms are generated from the grain-flipping probability model in building the magnetized medium [51]. This model generates realistic 2D waveforms with media noise. A description of the waveforms is provided in Fig. 6.6 where 25 different reader positions separated by $1/8$ of the track width (track pitch (TP)) provide 25 different readback waveforms mainly from the 3 middle tracks. Also, the ADC's sample at two times the bit rate, $(2/T)$, that means there are two ADC samples for every user bit. Consequently, the equalizers described in this chapter become fractionally spaced equalizers (FSE). The only change to the equations described earlier is with the vector of readback samples in (6.23) that changes

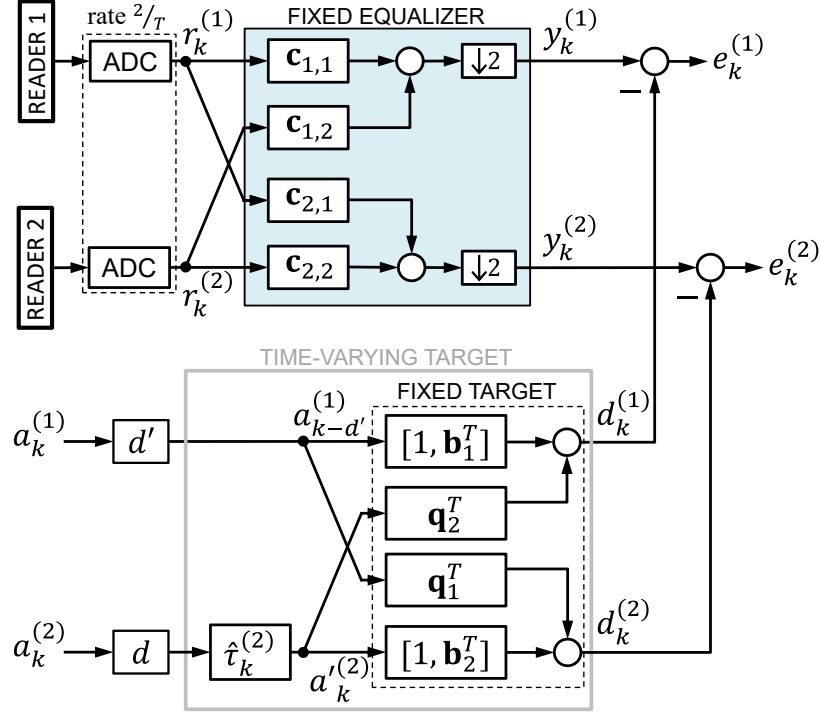


Figure 6.7: The alternative strategy 2 for equalizing $N = 2$ readback waveforms shown in Fig. 6.6 covering track 1 with $\Delta T_1 = 0$ and track 2 with $\Delta T_2/T = 2 \times 10^{-4}$.

to $\mathbf{r}_k = [\mathbf{r}_{2k}, \dots, \mathbf{r}_{2k-Nc+1}]^T$.

There were two sets of waveforms provided. In the first set of waveforms, a write frequency offset with parameter $\Delta T_2/T = 2 \times 10^{-4}$ is injected only to the bits of track 2 while the rest of the tracks have the same bit positions. As illustrated in Fig. 6.6, track 2 exhibits wider bit period, $T_2 = T + \Delta T_2$, than the rest of the tracks with $T_0 = T_1 = T_3 = T_4 = T$, where T is twice the sampling period of the ADC's. Since the ADC's sample slightly faster than twice the bit rate of track 2, the last ADC sample at the end of the sector of length $L = 41206$ bits, is $L\Delta T_2 = 8.24$ bits ahead of the actual position of the last bit on track 2.

The second set of 25 waveforms are also generated according to Fig. 6.6 without injecting any timing offset to the bits of track 2. Therefore all tracks have the same bit periods, $T_0 = T_1 = T_2 = T_3 = T_4 = T$. We refer to the first and second sets of waveforms, as the asynchronous and synchronous waveforms, respectively. Also, the bit-aspect ratio

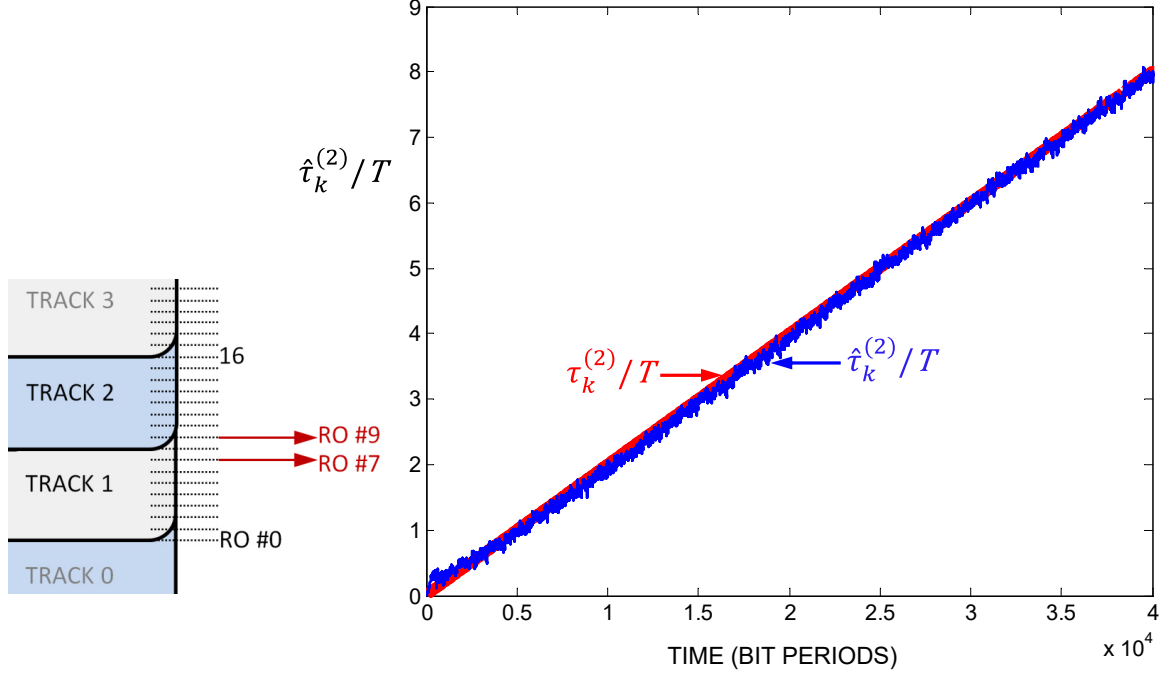


Figure 6.8: The PLL performance in estimating the timing offset of track 2.

defined as the ratio of the track pitch to the bit length (BL) is 3.

We consider first the alternative strategy 2 with the LMS algorithm, according to (6.29), (6.13), and (6.31). Fig. 6.7 illustrates this strategy for the case where $\Delta T_1/T = 0$ and $\Delta T_2/T = 2 \times 10^{-4}$. Since $\Delta T_1/T = 0$, as Fig. 6.7 shows, the corresponding interpolation filter reduces to a simple delay element of d' that is the delay of the equalizer cascade.

Fig. 6.8 plots an example of the second-order PLL performance in realization of Fig. 6.7, when the user bits were known and when the two readers at positions #7 and #9 were selected. The number of FSE coefficients was $N_c = 17$, the target memory $\mu = 1$, the delay elements were $d' = 4$ and $d = -46$, and the length of the interpolation filter was $N_s = 101$. The PLL parameters were $\alpha = 0.01$ and $\beta = \alpha^2/4$. Also the LMS step size $\nu = 0.001$. We observe that the PLL can effectively track the actual timing of track 2.

Nevertheless, a critically important performance criteria is the convergence rate of the adaptive algorithm that computes the equalizer and the target pair, since in practice, only a short preamble of a few hundred bits are available for computing the equalizer and the target pair. Fig. 6.9 compares the convergence of the LMS versus the RLS algorithm during the

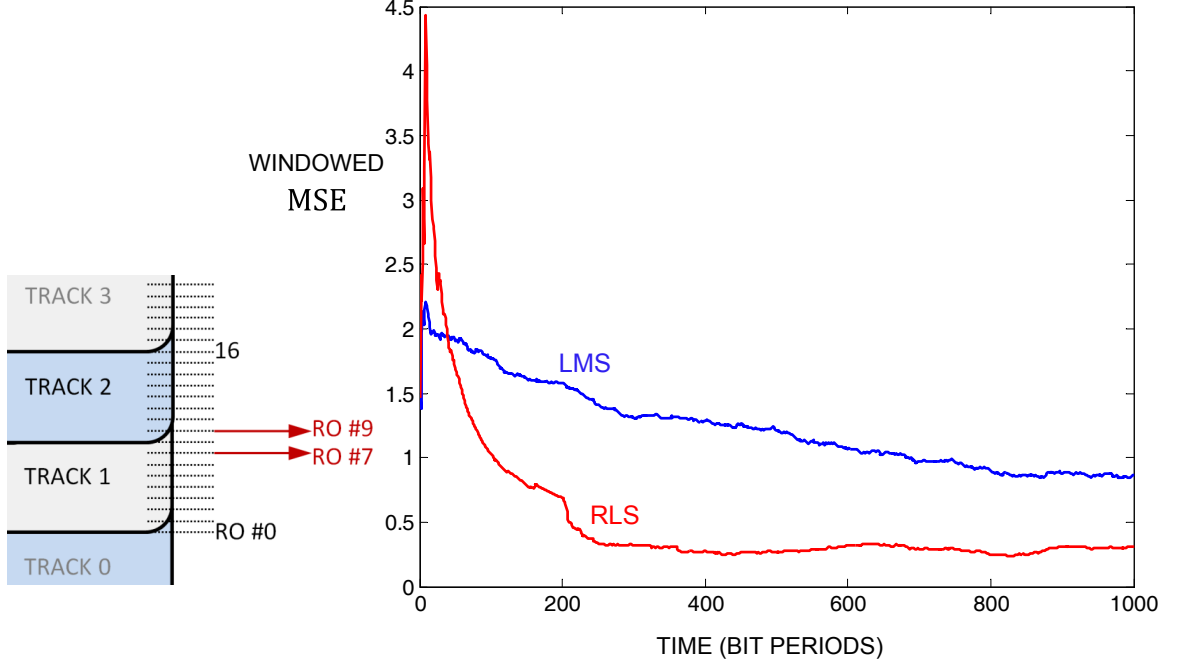


Figure 6.9: Convergence of the LMS versus RLS algorithms when used in the alternative strategy 2 of Fig. 6.7. MSE is averaged over the past 100 samples.

first 1000 bits of the sector in realization of Fig. 6.7. The figure plots the moving average over the past 100 samples of the MSE measure in (6.27) for the LMS algorithm and the WMSE measure in (6.32) for the RLS algorithm. The simulation environment is set similar to Fig. 6.8 with additional parameters $\lambda = 1$ and $\delta = 1.5$ for the RLS algorithm. As expected, we observe that the RLS algorithm is extremely faster than the LMS algorithm. The RLS converges almost at the 200-th bit while the LMS algorithm still continues to converge beyond 1000-th bit.

Based on this observation, we next consider only the RLS algorithm for computing the equalizer and the target pair during a preamble of the first 250 bits of the sector. We use the remaining length of the sector for testing the performance of our proposed equalization strategy followed by the ROTAR algorithm.

Fig. 6.10 shows the proposed equalizer performance over a period in the middle of the sector. The solid signal is the equalizer output $y_k^{(2)}$ and the dashed signal is the corresponding signal $d_k^{(2)}$ from the target branch in Fig. 6.7. The good match between these two signals

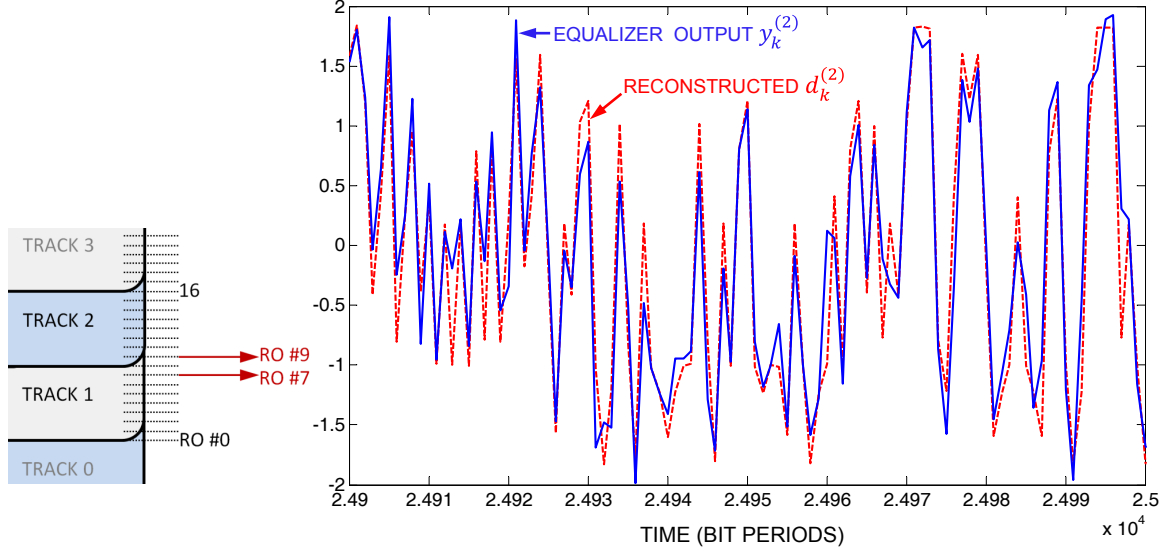


Figure 6.10: The MIMO equalizer output $y_k^{(2)}$ versus the corresponding signal $d_k^{(2)}$ from the target branch in Fig. 6.7.

verifies the effectiveness of the proposed equalization strategy.

Nonetheless, the ultimate performance measure of an equalization strategy is the final BER performance after the following detector. To this end, we combine the proposed equalization strategy in Algorithm 6 with the ROTAR algorithm as illustrated in Fig. 6.11.

The BER performance of the ultimate proposed read channel is shown and compared with three other read channels in Fig. 6.12. The figure plots the average BER performance for the two tracks being detected for different readers selected. For example, $TP/8$ is the distance between the two readers selected at positions #7 and #9, $TP/4$ refers to the readers at positions #6 and #10, and so on. Note that the larger the distance between the two readers, the less ITI exists in the readback waveforms.

The curve labeled “OFFSET, MIMO+ROTAR” shows the performance of the proposed read channel where the equalization of Fig. 6.7 is followed by a 16-state ROTAR with memory parameters $M_1 = 0$ and $M_2 = 2$, and PSP for timing estimation. Here, all simulation parameters are set as before in Fig. 6.10.

The curve labeled “OFFSET, 2 MISO+VITERBI” shows the performance of a conventional read channel that separately detects the two tracks using a pair of two MISO

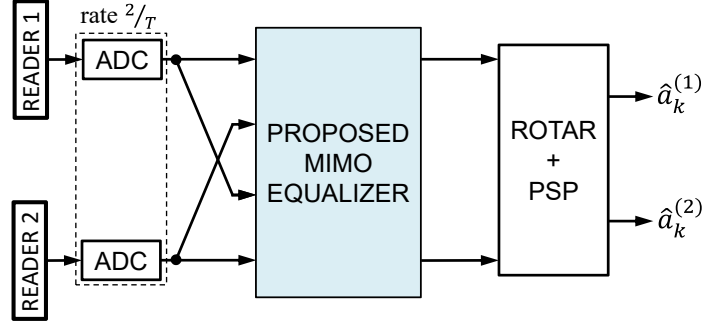


Figure 6.11: The proposed read channel for jointly detecting the two tracks of Fig. 6.1.

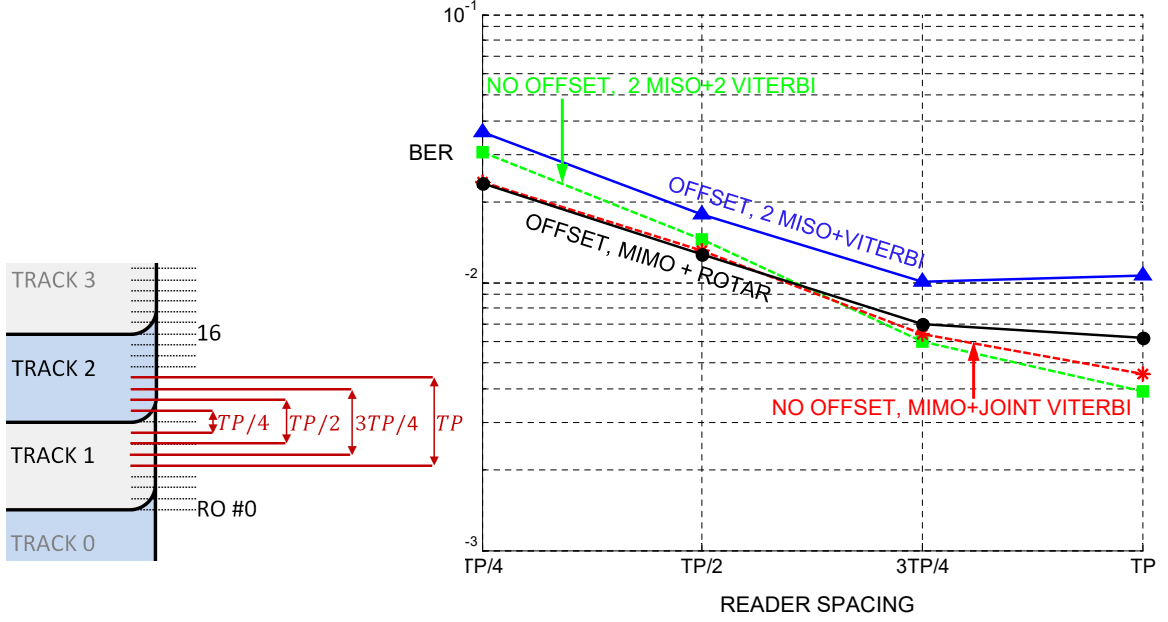


Figure 6.12: The BER performance of the proposed read channel.

equalizers followed by two independent Viterbi detectors. In particular, to detect track 1 with $\Delta T_1 = 0$, a conventional GPR equalization strategy is used where the equalizer and the target pair are computed using a RLS algorithm. This strategy uses Algorithm 6 where line 1 and line 11 are excluded, and where the interpolation filter in line 6 is simply replaced by a delay element of $d' = 4$. The equalization is followed by a 2-state Viterbi detector. Also, to detect track 2 with $\Delta T_2 \neq 0$, Algorithm 6 is used exactly and is followed by a 2-state Viterbi detector with PSP.

The figure shows that at least at this particular example, the proposed read channel outperforms the conventional read channel over the different readers' spacing selected.

Therefore, the figure verifies the validity of the proposed equalization strategy.

We also show in Fig. 6.12, the performance over the second set of synchronous waveforms, represented with dashed curves. The curve labeled “NO OFFSET, MIMO+JOINT VITERBI” shows the performance of a MIMO equalizer followed by a joint Viterbi detector for jointly detecting the two tracks of interest. In particular, a conventional GPR strategy is used where a RLS algorithm computes the equalizer and the target pair, according to Algorithm 6 where line 1 and line 11 are excluded, and where the both interpolation filters in line 6 are replaced by a delay element of $d' = 4$.

Lastly, the curve labeled “NO OFFSET, 2 MISO+2 VITERBI” shows the performance of the conventional read channel that separately detects the two tracks. In particular, since both tracks are synchronous to ADC’s sampling times, for detecting each track, the equalizers are computed using Algorithm 6 excluding line 1 and line 11, and with the interpolation filter in line 6 replaced by a delay element of $d' = 4$. After both MISO equalizers, two independent Viterbi detectors separately detected the two tracks.

We observe that the proposed read channel labeled “OFFSET, MIMO+ROTAR” tested on asynchronous waveforms is performing very close to the two read channels tested on synchronous waveforms. Here, we avoid a more exact comparison because we believe that the set of asynchronous waveforms include more media noise than the set of synchronous waveforms. Albeit, this close performance validates the proposed equalization strategy to be followed by the ROTAR algorithm.

6.4 Summary

We completed a proposed read channel for future generation of TDMR by proposing an equalization strategy to precede the ROTAR detector of chapter 5. The proposed equalization strategy equalizes asynchronous ADC outputs to a *time-varying target* using a GPR approach where an adaptive algorithm updates the equalizer filters and the target while a second-order PLL adapts the timing offsets.

We verified the proposed MIMO equalizer using a semi-realistic data set that exhibits nonlinear channel response and dominant media noise. We also verified and compared the performance of the complete proposed read channel with three other read channels.

CHAPTER 7

CONCLUSION

This thesis proposes synchronization and detection algorithms for current and future generations of read channels for TDMR. Synchronization is a critical component of every read channel since knowing *where* the bits are is a prerequisite to detect *what* the bits are. Throughout the magnetic recording literature, synchronization is a moderately studied problem for current generations of TDMR read channel that detects a single track of interest at a time. In this context, this thesis proposes to change a conventional read channel architecture by switching the order in which synchronization is done *before* equalization, and instead synchronize *after* equalization, in order to significantly reduce implementation cost.

To the contrary, in a multitrack detection setting, there is no synchronization solution proposed when a few tracks are to be *jointly* detected from a few readback waveforms. This thesis provides a solution for synchronization in multitrack scenario for future generations of TDMR read channels. In this chapter, we summarize the main contributions of this thesis followed by suggestions for future works that can extend our work.

7.1 Contributions

1. In Chapter 3, we presented two strategies for mitigating ITI, one based on linear suppression and one based on soft cancellation. Numerical results demonstrated that regardless of the media noise, the relative advantage of the two detection strategies depends on the location of the track being detected. For the inner tracks near the center of the read-head array, the gain of the soft ITI cancellation detector over the linear detector is modest. For the outer tracks near the edge of the array, in contrast, the gain of the soft ITI cancellation detector over the linear detector is significant.

Therefore, for a given pass of a read-head array, the soft ITI cancellation strategy has demonstrated its ability to reliably recover data from more tracks than would otherwise be possible using linear ITI suppression.

2. In Chapter 4, we considered the problem of single-track detection, from multiple readback waveforms when the tracks have different frequency and phase offsets. First, in Sect. 4.1, we applied the proposed soft ITI cancellation detector of Chapter 3 to *asynchronously* mitigate ITI before synchronizing for and detecting every track of interest. Since the proposed detector breaks the problem down to several 1D detection problems, a soft-output Viterbi detector plus PSP for timing recovery was used for synchronization and detection of every track of interest. Numerical results showed that the proposed architecture is capable in mitigating ITI for the separable MIMO model considered.

Next in Sect. 4.2, we considered a realistic nonseparable MIMO model for TDMR channel. We discovered that, within working precision, the solution for joint optimization of the equalizer filters and the target is independent of timing offsets. This finding has important implication: As opposed to the conventional TDMR read channels where equalization is done *after* synchronization and therefore there is one synchronization block required for every reader used, we can effectively equalize *before* synchronizing to a track of interest and detect, and therefore we only need one synchronization block for detecting every track of interest, regardless of the number of readers used. Therefore, the contribution here was a significant decrease in the computational complexity of the read channel.

3. In Chapter 5, we proposed the ROTAR algorithm for multitrack detection of asynchronous tracks using multiple readback waveforms. ROTAR applies the Viterbi algorithm to a time-varying rotating target that absorbs the asynchrony of the multiple tracks being detected. Since a time-varying target can increase the number of trellis

states, to keep complexity low, ROTAR decomposes the timing offsets into their integer and fractional parts, and only uses the fractional parts to shift the target. As a result the target *rotates* around its original format. A further reduction in complexity is achieved by locking the ADC's to one track. ROTAR can employ per-survivor processing to estimate the unknown timings inside the joint Viterbi detector.

4. In Chapter 6, we proposed an equalization strategy to precede the ROTAR detector and completed the read channel. The proposed equalizer equalizes asynchronous ADC outputs to a *time-varying target* using a GPR approach where the equalizer filters and the target are computed using an RLS algorithm, while a second-order PLL adapts the unknown timing offsets.

We verified the proposed MIMO equalizer on a semi-realistic data set that exhibits nonlinear channel response and dominant media noise. We also verified and compared the performance of the complete proposed read channel with three other read channels.

7.2 Future Directions

7.2.1 Multi-Track Timing Error Detector

Throughout this thesis we have used M&M TED equation [23] wherever a PLL was used for timing recovery. M&M TED is originally derived for the simple case of synchronizing a single waveform to unknown timings and therein it fits within a 1D detection problem. In the single-track detection setting, the read channel breaks the problem down into one or several 1D problems where the well established synchronization strategies with 1D TED can be exploited. Therefore, M&M TED was suitable in Chapters 3 and 4.

Nevertheless, for joint detection of multiple tracks with different timings, a multitrack extension of M&M or any other TED is required. Although a two-dimensional extension of M&M TED is already proposed in [30], we should distinguish between a 2D TED and a multitrack TED. A 2D TED applies within a 2D channel model where also a 2D model

for timing offset is considered. As discussed in Chapter 2, a 2D channel model requires a large scan of the disk as opposed to our MIMO model where only a few number of readback waveforms are used for detecting multiple tracks. Also a 2D TED operates on a 2D model for timing offset, that is the 2D image includes timing uncertainty in both downtrack and crosstrack dimensions. To the contrary, in our MIMO model, we only consider timing offsets in the downtrack dimension. In this sense, a multitrack TED is an extension of 1D TED that includes the cross-talk or the ITI in estimating the timing error of each of the tracks.

In ROTAR algorithm of Chapter 5, Algorithm 5, line 14, we used *a* multitrack extension of original 1D M&M TED, according to:

$$\hat{\epsilon}_k^{(j)}(q) = \sum_{i=1}^N r_k^{(i)} \hat{o}_{k-1}^{(i,j)}(q) - r_{k-1}^{(i)} \hat{o}_k^{(i,j)}(q), \quad (7.1)$$

where $\hat{\epsilon}_k^{(j)}(q)$ is the estimated timing error of track j at ending state q and time k , $r_k^{(i)}$ is the readback i sample at time k , and $\hat{o}_k^{(i,j)}(q)$ is the expected output from track j to reader i on the survivor path ending at state q .

The above equation implements an idea that the estimated error of every track should be the summation of the estimate that each readback waveform provides for that track. Even though, the above equation has performed well in our simulations, incurring no loss in the final BER performance, it is only an empirical equation. Therefore, a derivation of multitrack extension of M&M TED, for example, for use in multitrack settings is very relevant.

7.2.2 Media Noise Mitigation for ROTAR

As stated in Chapter 2, the data dependent media noise forms 80% – 90% of the total noise in magnetic recording channel. Pattern-dependent noise-predictive strategies [4] use the colored feature of the media noise to predict and subtract it from the input signal to the detector. Here, the media noise is modeled as an auto-regressive Gaussian process where

each noise sample is modeled as a weighted sum of its previous samples plus AWGN. Therefore, an adaptive algorithm, for example LMS, can compute the coefficients of the noise model. On the other hand, since these coefficients are data-dependent, each transition in a trellis detector specifies a noise sample of its own. Therefore, the trellis should be set up in such a way that the detector can determine the input bit as well as the predicted noise sample. The implication is that if the ISI memory is μ and the length of the noise predictor is Δ , then the number of trellis states will at least be $2^{\mu+\Delta}$, that is a notable increase in the detector complexity.

On the other hand, as explained in Chapter 5, the ROTAR algorithm also increases the number of trellis states by a factor of $\prod_{j=1}^K 2^{M_j}$, where K is the number of tracks to be detected, and M_j is the extra memory parameter for every track j . Consequently, before applying PDNP to the ROTAR detector, a strategy to reduce the overall complexity is needed.

Appendices

APPENDIX A

DERIVATION OF (3.6) FOR LINEAR CASE

For the case of linear ITI suppression of (3.4), the mean-squared error of (3.5) can be written as

$$\begin{aligned}
 \text{MSE} &= E \left(\left(\mathbf{w}_i^T \mathbf{r}_k - \left(x_k^{(i)} + m_k^{(i)} \right) \right)^2 \right) \\
 &= \mathbf{w}_i^T E \left(\mathbf{r}_k \mathbf{r}_k^T \right) \mathbf{w}_i + E_x + 2\sigma_j^2 E_q \\
 &\quad - 2\mathbf{w}_i^T E \left(\left(x_k^{(i)} + m_k^{(i)} \right) \mathbf{r}_k \right) \\
 &= \mathbf{w}_i^T \mathbf{R} \mathbf{w}_i + E_x + 2\sigma_j^2 E_q - 2\mathbf{w}_i^T \mathbf{p} \tag{A.1}
 \end{aligned}$$

$$\begin{aligned}
 &= (\mathbf{w}_i - \mathbf{R}^{-1} \mathbf{p})^T \mathbf{R} (\mathbf{w}_i - \mathbf{R}^{-1} \mathbf{p}) + E_x \\
 &\quad + 2\sigma_j^2 E_q - \mathbf{p}^T \mathbf{R}^{-1} \mathbf{p} \tag{A.2}
 \end{aligned}$$

where we have introduced the correlation matrix $\mathbf{R} = E(\mathbf{r}_k \mathbf{r}_k^T)$ and correlation vector $\mathbf{p} = E((x_k^{(i)} + m_k^{(i)}) \mathbf{r}_k)$, and where the last equality follows from completing the square. From (3.3) we can compute \mathbf{R} and \mathbf{p} as

$$\begin{aligned}
 \mathbf{R} &= (E_x + 2\sigma_j^2 E_q) \sum_n \mathbf{g}_n \mathbf{g}_n^T + \left(\frac{N_0}{2T} \right) \mathbf{I}, \text{ and} \\
 \mathbf{p} &= (E_x + 2\sigma_j^2 E_q) \mathbf{g}_i. \tag{A.3}
 \end{aligned}$$

Since \mathbf{R} is symmetric, the quadratic form for the first term in (A.2) implies that it cannot be negative. We can thus minimize MSE by forcing the first term to zero, which is achieved when $\mathbf{w}_i = \mathbf{R}^{-1} \mathbf{p}$, which reduces to (3.6) when using the results of (A.3), along with the identity $M_0 = 4\sigma_j^2 E_q$.

APPENDIX B

DERIVATION OF (3.16) FOR SOFT ITI CANCELLATION

The defining equation of (11) for the soft ITI cancellation scheme can be rewritten as

$$y_k^{(i)} = \mathbf{w}_i^T \tilde{\mathbf{r}}_k \quad (\text{B.1})$$

where we have introduced $\tilde{\mathbf{r}}_k$ the residual observation vector after soft cancellation, namely

$$\tilde{\mathbf{r}}_k = \mathbf{r}_k - \sum_{n \in \mathcal{P}} \tilde{x}_n^{(n)} \mathbf{g}_n \quad (\text{B.2})$$

Therefore, following the same procedure as in Appendix A, we can write the MSE after soft ITI cancellation as

$$\text{MSE} = \left(\mathbf{w}_i - \tilde{\mathbf{R}}^{-1} \mathbf{p} \right)^T \tilde{\mathbf{R}} \left(\mathbf{w}_i - \tilde{\mathbf{R}}^{-1} \mathbf{p} \right) + E_x + 2\sigma_j^2 E_q - \mathbf{p}^T \tilde{\mathbf{R}}^{-1} \mathbf{p}, \quad (\text{B.3})$$

where from (5.3) and (B.2) we have

$$\begin{aligned} \mathbf{p} &= E \left(\left(x_k^{(i)} + m_k^{(i)} \right) \tilde{\mathbf{r}}_k \right) \\ &= E \left(\left(x_k^{(i)} + m_k^{(i)} \right) \mathbf{r}_k \right) \\ &= (E_x + 2\sigma_j^2 E_q) \mathbf{g}_i \end{aligned} \quad (\text{B.4})$$

and

$$\begin{aligned}
\tilde{\mathbf{R}} &= E(\tilde{\mathbf{r}}_k \tilde{\mathbf{r}}_k^T) \\
&= E_x \sum_{n \notin \mathcal{P}} \mathbf{g}_n \mathbf{g}_n^T + \sum_{n \in \mathcal{P}} E \left(\left(x_k^{(n)} - \tilde{x}_k^{(n)} \right)^2 \right) \mathbf{g}_n \mathbf{g}_n^T \\
&\quad + 2\sigma_j^2 E_q \sum_n \mathbf{g}_n \mathbf{g}_n^T + \left(\frac{N_0}{2T} \right) \mathbf{I} \\
&= E_x \sum_n \alpha_n \mathbf{g}_n \mathbf{g}_n^T + 2\sigma_j^2 E_q \sum_n \mathbf{g}_n \mathbf{g}_n^T + \left(\frac{N_0}{2T} \right) \mathbf{I}, \tag{B.5}
\end{aligned}$$

where we have introduced

$$\alpha_n = \begin{cases} 1, & \text{for } n \notin \mathcal{P} \\ E((x_k^{(n)} - \tilde{x}_k^{(n)})^2)/E_x, & \text{for } n \in \mathcal{P} \end{cases}. \tag{B.6}$$

We can thus minimize MSE by forcing the first term in (B.3) to zero, which is achieved when $\mathbf{w}_i = \tilde{\mathbf{R}}^{-1} \mathbf{p}$. This reduces to (3.16) when using (B.4) and (B.5).

APPENDIX C

JOINT OPTIMIZATION OF TARGET AND FSE, IN SINGLE-TRACK DETECTION

In the following, we derive the MMSE solution for jointly optimizing N equalizer filters for N readback waveforms, and a vector-valued target, to be used in single-track detection.

Fig. C.1 shows an example of $N = 3$ equalizer filters and a vector-valued target. Here we consider the case when the ADC's are sampling at twice the bit rate, therefore, each equalizer filter has two coefficients per bit period, i.e. each equalizer filter is a fractionally-spaced equalizer (FSE).

Let $r_\ell^{(i)}$ denote the ℓ -th sample of the i -th readback waveform for $i \in \{1, \dots, N\}$ sampled at rate $2/T$ or twice the bit rate. Combining the N samples at time ℓ into a $N \times 1$ vector yields:

$$\mathbf{r}_\ell = \begin{bmatrix} r_\ell^{(1)} \\ \vdots \\ r_\ell^{(N)} \end{bmatrix}. \quad (\text{C.1})$$

We define each equalizer filter i with N_c coefficients, according to $\mathbf{c}^{(i)} = [c_0^{(i)}, c_1^{(i)}, \dots,$

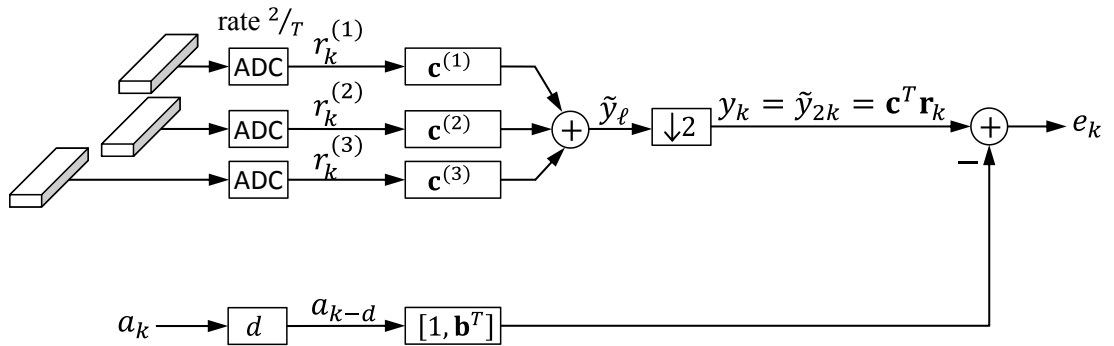


Figure C.1: Joint optimization of target and equalizer for the case of $N = 3$ readers and $K = 1$ track of interest.

$c_{Nc-1}^{(i)}]^T$. Combining the N coefficients across N filters at time ℓ into a $N \times 1$ vector yields:

$$\mathbf{c}_\ell = \begin{bmatrix} c_\ell^{(1)} \\ \vdots \\ c_\ell^{(N)} \end{bmatrix}. \quad (\text{C.2})$$

The N equalizer filter outputs are added and downsampled, so that the bank of N filters can be viewed as an N -input single-output FSE with coefficients $\{\mathbf{c}_0, \dots, \mathbf{c}_{Nc-1}\}$, whose k -th output can be written as $y_k = \underline{\mathbf{c}}^T \underline{\mathbf{r}}_k$, where

$$\underline{\mathbf{c}} = \begin{bmatrix} \mathbf{c}_0 \\ \vdots \\ \mathbf{c}_{Nc-1} \end{bmatrix}_{NNc \times 1} \quad \text{and} \quad \underline{\mathbf{r}}_k = \begin{bmatrix} \mathbf{r}_{2k} \\ \vdots \\ \mathbf{r}_{2k-Nc+1} \end{bmatrix}_{NNc \times 1}. \quad (\text{C.3})$$

We consider target to be monic and of the form $[1, \mathbf{b}^T]$, where $\mathbf{b} = [b_1, \dots, b_\mu]^T$ is the target tail and μ is the target memory. Filtering the user bits by the target yields the signal $a_{k-d} + \mathbf{b}^T \mathbf{a}_k$, where d is the delay parameter of the equalizer cascade, and where $\mathbf{a}_k = [a_{k-d-1}, \dots, a_{k-d-\mu}]^T$. With this terminology, the optimization problem is to jointly choose the equalizer $\underline{\mathbf{c}}$ and the target \mathbf{b} to minimize the mean-squared error $MSE = E(e_k^2)$ where

$$e_k = \underline{\mathbf{c}}^T \underline{\mathbf{r}}_k - a_{k-d} - \mathbf{b}^T \mathbf{a}_k. \quad (\text{C.4})$$

The error can be simplified by cascading the FSE coefficients and the target into a single vector \mathbf{w} , and also cascading the inputs to the FSE filters and the user bits into a single vector \mathbf{v}_k , so that e_k becomes:

$$e_k = \mathbf{w}^T \mathbf{v}_k - a_{k-d}, \quad (\text{C.5})$$

where

$$\mathbf{w} = \begin{bmatrix} \underline{\mathbf{c}} \\ -\mathbf{b} \end{bmatrix}_{(NNc+\mu) \times 1} \quad \text{and} \quad \mathbf{v}_k = \begin{bmatrix} \underline{\mathbf{r}}_k \\ \mathbf{a}_k \end{bmatrix}_{NNc \times 1}. \quad (\text{C.6})$$

Therefore, the MSE is:

$$\begin{aligned}
MSE &= E((\mathbf{w}^T \mathbf{v}_k - a_{k-d})^2) \\
&= \mathbf{w}^T \mathbf{R}_{vv} \mathbf{w} + 1 - 2\mathbf{w}^T \mathbf{p} \\
&= (\mathbf{w}^T - \mathbf{R}_{vv}^{-1} \mathbf{p})^T \mathbf{R}_{vv} (\mathbf{w}^T - \mathbf{R}_{vv}^{-1} \mathbf{p}) + 1 - \mathbf{p}^T \mathbf{R}_{vv}^{-1} \mathbf{p}, \tag{C.7}
\end{aligned}$$

where the covariance matrix \mathbf{R}_{vv} and the cross-covariance vector \mathbf{p} respectively are:

$$\mathbf{R}_{vv} = E(\mathbf{v}_k \mathbf{v}_k^T) \quad \text{and} \quad \mathbf{p} = E(\mathbf{v}_k a_{k-d}). \tag{C.8}$$

\mathbf{w} only appears in the first term of equation (C.7). This term has quadratic form, and since \mathbf{R}_{vv} is symmetric, the term cannot be negative, and can only be minimized by setting it to zero. Therefore, the \mathbf{w} that minimizes the MSE in equation (C.7) is:

$$\mathbf{w} = \mathbf{R}_{vv}^{-1} \mathbf{p}. \tag{C.9}$$

APPENDIX D

JOINT OPTIMIZATION OF TARGET AND FSE, IN MULTITRACK DETECTION

In the following, we derive the MMSE solution for jointly optimizing $K \times N$ equalizer filters for N readback waveforms, and K matrix-valued targets, prior to multitrack detection of the K tracks of interest.

Fig. D.1 shows an example of $K \times N = 2 \times 2 = 4$ equalizer filters prior to a multitrack detection of the two tracks. Here we consider the case when the ADC's are sampling twice the bit rate, therefore, each equalizer filter has two coefficients per bit period, i.e. each equalizer filter is a fractionally-spaced equalizer (FSE).

Let $r_\ell^{(i)}$ denote the ℓ -th sample of the i -th readback waveform sampled at rate $2/T$ or twice the bit rate. Combining the N samples at time ℓ into a $N \times 1$ vector yields:

$$\mathbf{r}_\ell = \begin{bmatrix} r_\ell^{(1)} \\ \vdots \\ r_\ell^{(N)} \end{bmatrix}. \quad (\text{D.1})$$

We define each equalizer filter $j, i \in \{1, \dots, K\}, \{1, \dots, N\}$ with N_c coefficients, according to $\mathbf{c}_{j,i} = [c_0^{(j,i)}, c_1^{(j,i)}, \dots, c_{N_c-1}^{(j,i)}]^T$. Combining the N coefficients of the j -th bank of N filters at time ℓ into a $N \times 1$ vector yields:

$$\mathbf{c}_\ell^{(j)} = \begin{bmatrix} c_\ell^{(j,1)} \\ \vdots \\ c_\ell^{(j,N)} \end{bmatrix}. \quad (\text{D.2})$$

The N equalizer filter outputs of the j -th bank are added and downsampled, so that the

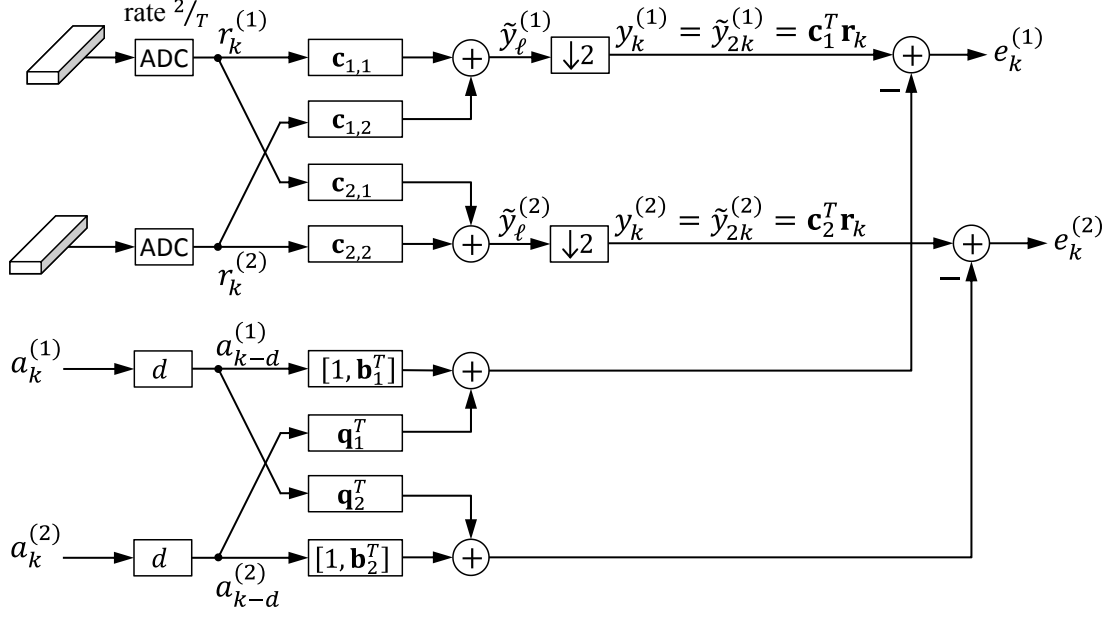


Figure D.1: Joint optimization of target and equalizer for the case of $N = 2$ readers and $K = 2$ tracks of interest.

j -th bank of N filters can be viewed as an N -input single-output FSE with coefficients $\{\mathbf{c}_0^{(j)}, \dots, \mathbf{c}_{Nc-1}^{(j)}\}$, whose k -th output can be written as $y_k^{(j)} = \underline{\mathbf{c}}_j^T \underline{\mathbf{r}}_k$, where

$$\underline{\mathbf{c}}_j = \begin{bmatrix} \mathbf{c}_0^{(j)} \\ \vdots \\ \mathbf{c}_{Nc-1}^{(j)} \end{bmatrix}_{NNc \times 1} \quad \text{and} \quad \underline{\mathbf{r}}_k = \begin{bmatrix} \mathbf{r}_{2k} \\ \vdots \\ \mathbf{r}_{2k-Nc+1} \end{bmatrix}_{NNc \times 1}. \quad (\text{D.3})$$

Since there are K tracks, we have K matrix-valued targets. We consider each target j to have the form

$$\begin{bmatrix} 1 & \mathbf{b}_j^T \\ \mathbf{q}_{\ell_1}^T \\ \vdots \\ \mathbf{q}_{\ell_{K-1}}^T \end{bmatrix}_{K \times (\mu+1)}, \quad [\ell_1, \dots, \ell_{K-1}] = [1, \dots, j-1, j+1, \dots, K], \quad (\text{D.4})$$

where the first row includes $\mathbf{b}_j = [b_1^{(j)}, \dots, b_\mu^{(j)}]^T$, the response tail from track j , and μ is

the response memory, and where the rest of the rows are the responses $\{\mathbf{q}_{\ell_m}\}$ from other $K - 1$ tracks. Filtering the user bits on all K tracks with their corresponding targets yields K signals, each of the form

$$s_k^{(j)} = a_{k-d}^{(j)} + \mathbf{b}_j^T \mathbf{a}_k^{(j)} + \sum_{m=1}^{K-1} \mathbf{q}_{\ell_m}^T \begin{bmatrix} a_{k-d}^{(\ell_m)} \\ \mathbf{a}_k^{(\ell_m)} \end{bmatrix}, [\ell_1, \dots, \ell_{K-1}] = [1, \dots, j-1, j+1, \dots, K], \quad (\text{D.5})$$

where $\mathbf{a}_k^{(j)} = [a_{k-d-1}^{(j)}, \dots, a_{k-d-\mu}^{(j)}]^T$, and where d is the delay parameter of the equalizer cascade. With this terminology, the optimization problem is to jointly choose the MIMO equalizer $\{\underline{\mathbf{c}}_j\}$ and the targets parameters $\{\mathbf{b}_j\}$ and $\{\mathbf{q}_j\}$, to minimize the mean-squared error

$$MSE = E\left(\sum_{j=1}^K e_k^{(j)2}\right), \quad (\text{D.6})$$

where

$$e_k^{(j)} = \underline{\mathbf{c}}_j^T \mathbf{r}_k - s_k^{(j)}. \quad (\text{D.7})$$

Each error can be simplified by cascading the corresponding FSE coefficients and the target into a single vector \mathbf{w}_j , and also cascading the corresponding inputs to the j -th FSE filters and the user bits into a single vector $\mathbf{v}_k^{(j)}$, so that $e_k^{(j)}$ becomes:

$$e_k^{(j)} = \mathbf{w}_j^T \mathbf{v}_k^{(j)} - a_{k-d}^{(j)}, \quad (\text{D.8})$$

where

$$\mathbf{w}_j = \begin{bmatrix} \underline{\mathbf{c}}_j \\ -\mathbf{b}_j \\ -\mathbf{q}_{\ell_1} \\ \vdots \\ -\mathbf{q}_{\ell_{K-1}} \end{bmatrix}_{(NNc+K\mu+K-1) \times 1} \quad \text{and} \quad \mathbf{v}_k^{(j)} = \begin{bmatrix} \underline{\mathbf{r}}_k \\ \mathbf{a}_k^{(j)} \\ \begin{bmatrix} a_{k-d}^{(\ell_1)} \\ \mathbf{a}_k^{(\ell_1)} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} a_{k-d}^{(\ell_{K-1})} \\ \mathbf{a}_k^{(\ell_{K-1})} \end{bmatrix} \end{bmatrix}_{(NNc+K\mu+K-1) \times 1}, \quad (\text{D.9})$$

where $[\ell_1, \dots, \ell_{K-1}] = [1, \dots, j-1, j+1, \dots, K]$. Therefore, the *MSE* is:

$$\begin{aligned} MSE &= E\left(\sum_{j=1}^K (\mathbf{w}_j^T \mathbf{v}_k^{(j)} - a_{k-d}^{(j)})^2\right) \\ &= \sum_{j=1}^K \mathbf{w}_j^T \mathbf{R}_{vv}^{(j)} \mathbf{w}_j + 1 - 2\mathbf{w}_j^T \mathbf{p}_j \\ &= \sum_{j=1}^K (\mathbf{w}_j^T - \mathbf{R}_{vv}^{(j)-1} \mathbf{p}_j)^T \mathbf{R}_{vv}^{(j)-1} (\mathbf{w}_j^T - \mathbf{R}_{vv}^{(j)-1} \mathbf{p}_j) + 1 - \mathbf{p}_j^T \mathbf{R}_{vv}^{(j)-1} \mathbf{p}_j, \end{aligned} \quad (\text{D.10})$$

where the covariance matrix $\mathbf{R}_{vv}^{(j)}$ and the cross-covariance vector \mathbf{p}_j , respectively are:

$$\mathbf{R}_{vv}^{(j)} = E(\mathbf{v}_k^{(j)} \mathbf{v}_k^{(j)T}) \quad \text{and} \quad \mathbf{p}_j = E(\mathbf{v}_k^{(j)} a_{k-d}^{(j)}). \quad (\text{D.11})$$

\mathbf{w}_j only appears in the first term in the sum of equations (D.10). This term has quadratic form, and since $\mathbf{R}_{vv}^{(j)}$ is symmetric, the term cannot be negative, and can only be minimized by setting it to zero. Therefore, the \mathbf{w}_j that minimizes the *MSE* in equation (D.10) is:

$$\mathbf{w}_j = \mathbf{R}_{vv}^{(j)-1} \mathbf{p}_j, \forall j \in \{1, \dots, K\}. \quad (\text{D.12})$$

REFERENCES

- [1] Hard disk drive, [Online]. Available: <https://flashdba.com/category/storage-for-dbas/storage-myths/> (visited on 11/2016).
- [2] Perpendicular recording, [Online]. Available: https://en.wikipedia.org/wiki/Perpendicular_recording (visited on 11/2016).
- [3] ASTC roadmap, [Online]. Available: http://idema.org/?page_id=5868 (visited on 11/2016).
- [4] B. Vasić and E. Kurtas, *Coding and signal processing for magnetic recording systems*. New York: CRC Press, 2004.
- [5] S. Dahandeh, M. F. Erden, and R. Wood, “Areal-density gains and technology roadmap for two-dimensional magnetic recording,” in *TMRC, 2015*, Aug. 2015, Paper F1.
- [6] R. Wood, M. Williams, A. Kavcic, and J. Miles, “The feasibility of magnetic recording at 10 terabits per square inch on conventional media,” *IEEE Transactions on Magnetics*, vol. 45, no. 2, pp. 917–923, Feb. 2009.
- [7] A. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, Apr. 1967.
- [8] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate,” *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.
- [9] E. Ordentlich and R. Roth, “On the computational complexity of 2D maximum-likelihood sequence detection,” Hewlett-Packard Labs, Palo Alto, CA, Tech. Rep. HPL-2006-69, 2006.
- [10] J. K. Nelson, A. C. Singer, and U. Madhow, “Multi-directional decision feedback for 2D equalization,” in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, May 2004, pp. IV–921–924.
- [11] M. Marrow and J. K. Wolf, “Iterative detection of 2-dimensional ISI channels,” in *Proceedings 2003 IEEE Information Theory Workshop*, Paris, Mar. 2003, pp. 131–134.

- [12] Y. Wu, J. A. O'Sullivan, N. Singla, and R. S. Indeck, "Iterative detection and decoding for separable two-dimensional intersymbol interference," *IEEE Transactions on Magnetics*, vol. 39, no. 4, pp. 2115–2210, 2003.
- [13] S. M. Khatami and B. Vasić, "Generalized belief propagation detector for TDMR microcell model," *IEEE Transactions on Magnetics*, vol. 49, no. 7, pp. 3699–3702, 2013.
- [14] T. Cheng, B. J. Belzer, and K. Sivakumar, "Row-column soft-decision feedback algorithm for two-dimensional intersymbol interference," *IEEE Signal Processing Letters*, vol. 14, no. 7, pp. 433–436, 2007.
- [15] Y. Chen, P. Njeim, T. Cheng, B. J. Belzer, and K. Sivakumar, "Iterative soft decision feedback zig-zag equalizer for 2d intersymbol interference channels," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 2, pp. 167–180, Feb. 2010.
- [16] Y. Chen and S. G. Srinivasa, "Joint self-iterating equalization and detection for two-dimensional intersymbol-interference channels," *IEEE Transactions on Communications*, vol. 61, no. 8, pp. 3219–3230, Aug. 2013.
- [17] G. Mathew, E. Hwang, J. Park, G. Garfunkel, and D. Hu, "Capacity advantage of array-reader-based magnetic recording (ARMR) for next generation hard disk drives," *IEEE Transactions on Magnetics*, vol. 50, no. 3, pp. 155–161, Mar. 2014.
- [18] E. Soljanin and C. N. Georgiades, "Multihead detection for multitrack recording channels," *IEEE Transactions on Information Theory*, vol. 44, no. 7, pp. 2988–2997, Nov. 1998.
- [19] L. M. M. Myint, P. Supnithi, and P. Tantaswadi, "An inter-track interference mitigation technique using partial ITI estimation in patterned media storage," *IEEE Transactions on Magnetics*, vol. 45, no. 10, pp. 3691–3694, Oct. 2009.
- [20] W. Chang and J. R. Cruz, "Inter-track interference mitigation for bit-patterned magnetic recording," *IEEE Transactions on Magnetics*, vol. 46, no. 11, pp. 3899–3908, Nov. 2010.
- [21] G. Han, Y. L. Guan, K. Cai, and K. S. Chan, "Asymmetric iterative multi-track detection for 2-D non-binary LDPC-coded magnetic recording," *IEEE Transactions on Magnetics*, vol. 49, no. 10, pp. 5215–5221, Oct. 2013.
- [22] J. W. M. Bergmans, *Digital baseband transmission and recording*. Kluwer Academic Publishers, 1996.
- [23] K. Mueller and M. Müller, "Timing recovery in digital synchronous data receivers," *IEEE Transactions on Communications*, vol. 24, no. 5, pp. 516–531, May 1976.

- [24] J. R. Barry, A. Kavcic, S. W. LeLaughlin, A. Nayak, and W. Zeng, "Iterative timing recovery," *IEEE Signal Processing Magazine*, vol. 21, no. 1, pp. 89–102, Jan. 2004.
- [25] A. Vanelli-Coralli, P. Salmi, S. Cioni, G. E. Corazza, and A. Polydoros, "A performance review of psp for joint phase/frequency and data estimation in future broadband satellite networks," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 12, pp. 2298–2309, Dec. 2001.
- [26] P. Kovintavewat, J. R. Barry, M. F. Erden, and E. Kurtas, "A new timing recovery architecture for fast convergence," in *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, May 2003, II–13–II–16 vol.2.
- [27] P. Kovintavewat and J. R. Barry, "Per-survivor timing recovery for uncoded partial-response channels," in *Proc. IEEE Int. Conference on Communications*, Paris, Jun. 2004, pp. 2715–2718.
- [28] P. Kovintavewat, J. R. Barry, M. F. Erden, and E. M. Kurtas, "Per-survivor iterative timing recovery for coded partial response channels," in *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, vol. 4, Nov. 2004, pp. 2604–2608.
- [29] R. Raheli, A. Polydoros, and C.-K. Tzou, "Per-survivor processing: A general approach to mlse in uncertain environments," *IEEE Transactions on Communications*, vol. 43, no. 2/3/4, pp. 354–364, Feb. 1995.
- [30] K. M. Whelan, F. Balado, N. J. Hurley, and G. C. M. Silvestre, "A two-dimensional extension of the mueller and müller timing error detector," *IEEE Signal Processing Letters*, vol. 14, no. 7, pp. 457–460, 2007.
- [31] B. P. Reddy, S. G. Srinivasa, and S. Dahandeh, "Timing recovery algorithms and architectures for 2-D magnetic recording systems," *IEEE Transactions on Magnetics*, vol. 51, no. 4, pp. 1–7, Apr. 2015.
- [32] E. Modak, B. P. Reddy, and S. G. Srinivasa, "Optimum timing interpolation algorithm for 2-d magnetic recording systems," in *2016 Twenty Second National Conference on Communication (NCC)*, 2016, pp. 1–6.
- [33] D. Kim, M. J. Narasimha, and D. C. Coc, "Design of optimal interpolation filter for symbol timing recovery," *IEEE Transactions on Communications*, vol. 45, no. 7, pp. 877–884, 1997.
- [34] R. Wood, R. Galbraith, and J. Coker, "2-D magnetic recording: Progress and evolution," *IEEE Transactions on Magnetics*, vol. 51, no. 4, pp. 1–7, Apr. 2015.

- [35] E. Banan Sadeghian and J. R. Barry, "Soft intertrack interference cancellation for two-dimensional magnetic recording," *IEEE Transactions on Magnetics*, vol. 51, no. 6, pp. 1–9, Jun. 2015.
- [36] R. Koetter, A. C. Singer, and M. Tüchler, "Turbo equalization," *IEEE Signal Processing Magazine*, vol. 21, no. 1, pp. 67–80, Jan. 2004.
- [37] A. J. Viterbi, "Very low rate convolution codes for maximum theoretical performance of spread-spectrum multiple-access channels," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 4, pp. 641–649, May 1990.
- [38] P. Patel and J. Holtzman, "Analysis of a simple successive interference cancellation scheme in a DS/CDMA system," *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 5, pp. 796–807, Jun. 1994.
- [39] W. Zha and S. D. Blostein, "Soft-decision successive interference cancellation CDMA receiver with amplitude averaging and robust to timing errors," in *Proc. IEEE Global Telecommunications Conference, 2001. GLOBECOM '01.*, vol. 2, 2001, pp. 743–747.
- [40] R. R. Lopes and J. R. Barry, "The soft-feedback equalizer for turbo equalization of highly dispersive channels," *IEEE Journal on Selected Areas in Communications*, vol. 54, no. 5, pp. 783–788, May 2006.
- [41] X. Wang and H. V. Poor, "Iterative (turbo) soft interference cancellation and decoding for coded CDMA," *IEEE Transactions on Communications*, vol. 47, no. 7, pp. 1046–1061, Jul. 1999.
- [42] J. Wang and S. Li, "Soft versus hard interference cancellation in MMSE OSIC MIMO detector: A comparative study," in *4th International Symposium on Wireless Communication Systems, 2007. ISWCS 2007.*, Oct. 2007, pp. 642–646.
- [43] J. Hagenauer, "The turbo principle: Tutorial introduction and state of the art," in *Proc. International Symposium on Turbo Codes and Related Topics*, Brest, France, 1997, pp. 1–11.
- [44] X. Hu. Progressive Edge Growth (PEG) software, [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/PEG_ECC.html (visited on 11/2016).
- [45] F. Gilbert, F. Kienle, and N. Wehn, "Low complexity stopping criteria for UMTS turbo-decoders," in *The 57th IEEE Semiannual Vehicular Technology Conference, 2003. VTC 2003-Spring.*, vol. 4, Apr. 2003, pp. 2376–2380.

- [46] Y. Kanai, Y. Jinbo, T. Tsukamoto, S. J. Greaves, K. Yoshida, and H. Muraoka, "Finite-element and micromagnetic modeling of write heads for shingled recording," *IEEE Transactions on Magnetics*, vol. 46, no. 3, pp. 715–721, Mar. 2010.
- [47] D. D. Falconer and F. R. Magee, "Adaptive channel memory truncation for maximum likelihood sequence estimation," *The Bell System Technical Journal*, vol. 52, no. 9, pp. 1541–1562, Nov. 1973.
- [48] J. Moon and W. Zeng, "Equalization for maximum likelihood detectors," *IEEE Transactions on Magnetics*, vol. 31, no. 2, pp. 1083–1088, Mar. 1995.
- [49] B. Vasić, M. Khatami, Y. Nakamura, Y. Okamoto, Y. Kanai, J. R. Barry, S. W. McLaughlin, and E. Banan Sadeghian, "A study of TDMR signal processing opportunities based on quasi-micromagnetic simulations," *IEEE Transactions on Magnetics*, vol. 51, no. 4, pp. 1–7, Apr. 2015.
- [50] E. Banan Sadeghian and J. R. Barry, "The rotating-target algorithm for jointly detecting asynchronous tracks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 9, pp. 2463–2469, 2016.
- [51] K. S. Chan, R. Radhakrishnan, K. Eason, M. R. Elidrissi, J. J. Miles, B. Vasić, and A. R. Krishnan, "Channel models and detectors for two-dimensional magnetic recording," *IEEE Transactions on Magnetics*, vol. 46, no. 3, pp. 804–811, Mar. 2010.

VITA

Elnaz Banan Sadeghian received the B.S. degree in electrical engineering from Shahid Beheshti University, Tehran, Iran, in 2005, and the M.S. degree in Biomedical Engineering from AmirKabir University of Technology, Tehran, Iran, in 2008. She is currently pursuing her Ph.D. degree in electrical engineering at the Georgia Institute of Technology, Atlanta, Georgia, USA. Her current research interests are in the area of signal processing and communication theory, including synchronization, detection, equalization, and coding as applied to magnetic recording channels.